MSE | MASTER OF SCIENCE IN ENGINEERING

OST Eastern Switzerland University of Applied Sciences

Claudio Bertozzi

Matching and Conflation of Open Government Data with OpenStreetMap Data

Matching and Conflation of All The Places Data with OpenStreetMap Data

Master's Thesis

OST – Eastern Switzerland University of Applied Sciences
Campus Rapperswil-Jona

Supervision

Prof. Stefan F. Keller

08.2025

Abstract

This thesis addresses the absence of a reliable, openly licensed, repeatable workflow for conflating authoritative POI from Swiss OGD sources with heterogeneous OSM data—a gap that increases duplication, staleness, and manual curation cost. The objective was to design and evaluate an auditable end-to-end pipeline (DiffedPlaces) that ingests brand and retailer feeds from ATP together with contemporaneous OSM extracts, generates spatially and semantically blocked candidate pairs for Switzerland (Oct 2024–Aug 2025), and resolves matches via a tunable rule-based scorer and a supervised machine learning (ML) RandomForestClassifier. Two golden datasets underpinned evaluation: a brand-focused Aldi Süd Switzerland subset (246 outlets) and a stratified multi-category random sample (200 POI). The evolved ML matcher achieved Precision 1.0000, Recall 0.9957 (F1 0.9978) on the brand subset and improved F1 on the heterogeneous sample while substantially lowering false positives versus the tuned rule-based approach. The resulting workflow delivers reproducible, high-precision conflation, reduces audit workload, and provides a transferable governance template for integrating additional authoritative OGD feeds into OSM with transparent quality controls.

Keywords: DiffedPlaces, All The Places (ATP), OpenStreetMap (OSM), Data Conflation, Geospatial Data Management, Point of Interest (POI), Random Forest Classifier, Automated Data Processing, Open Government Data (OGD).

Executive Summary

Public geospatial data depends on reconciliation between authoritative brand / retailer feeds and community-maintained map features. Fragmented, one-off imports have produced duplicate, stale, or diverging entries that erode trust and inflate manual curation workload. This project delivers a repeatable workflow that ingests dated Swiss snapshots from ATP alongside current OSM extracts, harmonises tagging into a taxonomy, and prepares high-recall candidate pairs through joint spatial and semantic blocking. Core constraints—transparent auditability, commodity hardware (⩽ 16 GB RAM), and extensibility—guided each design decision, yielding predictable quality. The approach emphasises explainability (structured similarity breakdowns), governance (immutable provenance metadata, deterministic reruns), and incremental adoption (drop-in matcher profiles without retooling downstream audit). By explicitly modelling both spatial proximity and semantic compatibility early, the workflow reduces noisy pair proliferation and focuses human review where residual ambiguity genuinely exists.

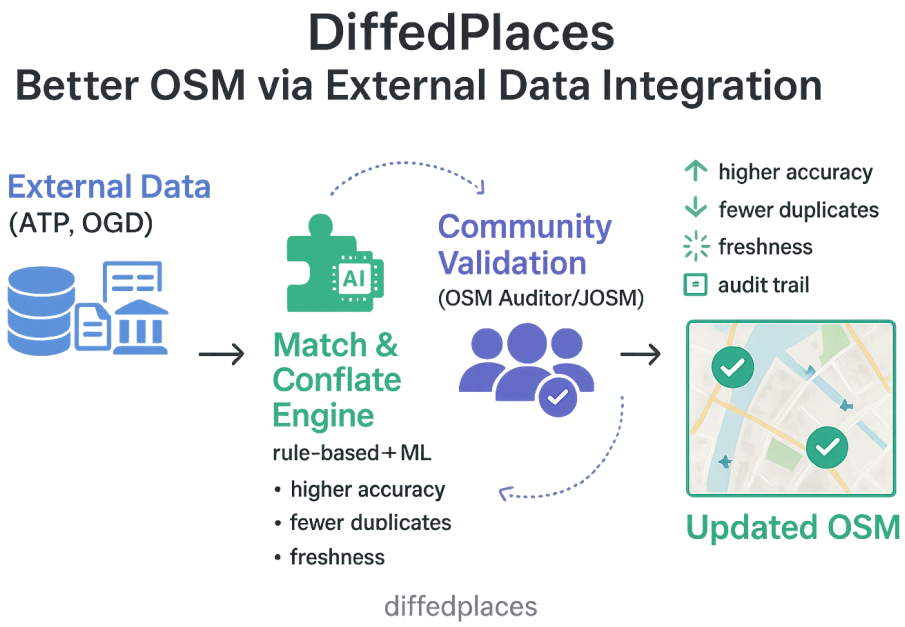

Figure 1: Early ideation diagram illustrating envisioned ingestion, matching, audit, and feedback loops.

The workflow operationalises two complementary matchers inside DiffedPlaces (i) a deterministic rules engine producing interpretable similarity decompositions and (ii) a supervised RandomForestClassifier (engineered lexical, spatial, semantic, structural features) trained with a staged regimen combining synthetic perturbations and high-precision pseudo-labels. Unified GeoJSON diffs progress through a multi-layer human-in-the-loop audit (command-line inspection, web-based collaborative review, quorum confirmation) before generating a ChangeXML batch for curator validation and upload. Idempotent ingestion, explicit provenance metadata, and reproducible configuration profiles minimise hidden state; semantic blocking sharply reduces implausible pair evaluations and thereby lowers false positives upstream of any machine learning decision boundary. Feature design balances robustness (token-set and character similarities, distance decay, tag agreement) with parsimony to support later optimisation and potential model distillation. The architecture isolates concerns (ingest, match, audit, publish) so that performance tuning or classifier upgrades do not ripple unpredictably into auditing or upload procedures.

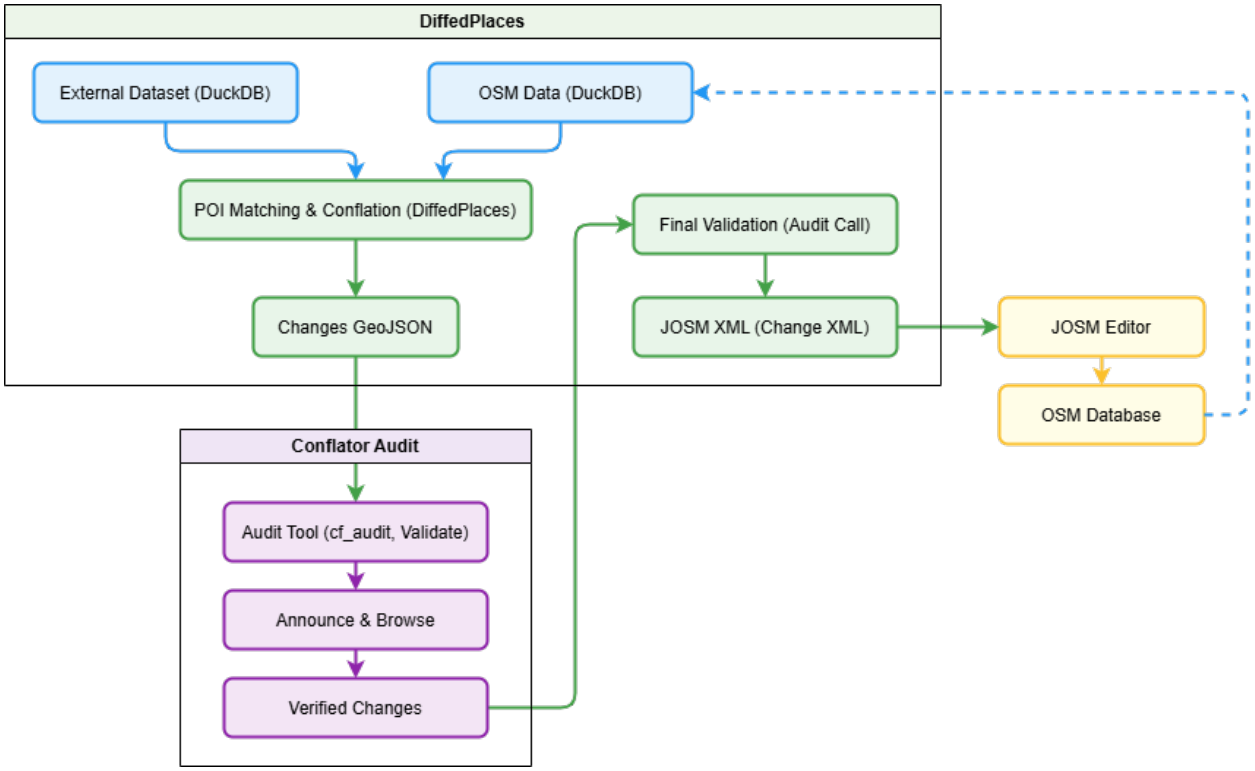

Figure 2: Operational pipeline: ingestion (blue), matching (green), auditing (purple), human upload (yellow).

Empirical validation on two golden datasets—a brand-focused Aldi Süd Switzerland subset (246 outlets) and a stratified multi-category sample (200 POIs)—shows the machine learning matcher attains near-zero false positives on the brand subset (Precision 1.0000, Recall 0.9957, F1 0.9978) and improves balanced performance on heterogeneous data (F1 0.8814 vs. 0.8587 for the tuned rule-based approach) while lowering False Positive Rate. These gains cut auditor review volume in dense urban clusters and provide a defensible precision baseline for integrating further OGDs feeds. Practical impact includes shorter publication lead times, higher mapper confidence in suggested merges, and clearer escalation paths for borderline cases. Remaining limitations include elevated inference latency relative to the rule-based path, limited golden coverage of rare categories, and absent automated drift detection. Proposed mitigations—feature pruning, alias expansion, adaptive thresholds, probability calibration, lightweight model distillation, and periodic drift health reports—outline a clear path to production hardening while preserving reproducibility and transparency.

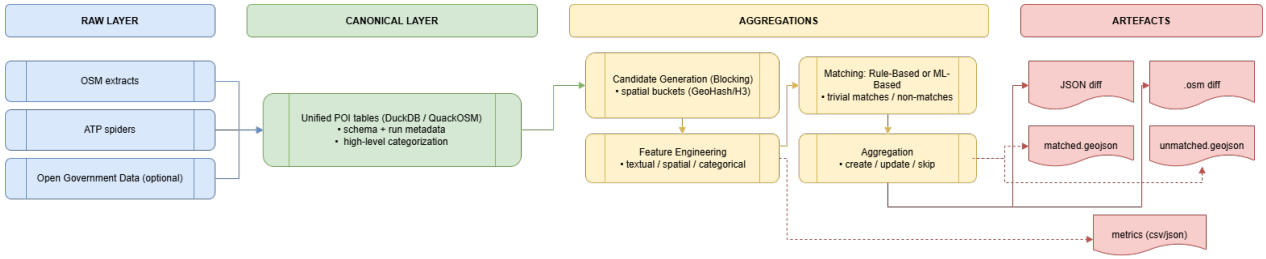

Figure 3: Data flow emphasising feedback loops from ingestion through audit back to refreshed snapshots.

Acknowledgment

I express sincere appreciation to Prof. Stefan F. Keller for his consistent guidance, critical feedback, and constructive challenge at each decision point. His ability to connect strategic direction with implementation detail materially elevated the rigor and practical relevance of this work.

Gratitude is extended to Sacha Brawer for incisive architectural discussions and for helping stress-test early assumptions around data ingestion, matcher extensibility, and reviewer ergonomics. His perspective helped sharpen the separation of concerns that underpins the reproducible workflow presented.

Another thanks go to Felix Reiniger of the IFS for hands-on support during iterative test runs and for engineering-focused collaboration while integrating auditing capabilities into the OSM Auditor environment.

Finally, thanks are given to all those who provided informal reviews or exploratory usage feedback during intermediate checkpoints; their observations helped refine clarity, robustness, and maintainability.

Contents

Chapter 1

Introduction

1.1 Problem Statement & Motivation

Despite the continuous growth of OGD portals, the OSM ecosystem still lacks a *reliable, repeatable, and openly licensed workflow* to conflate authoritative POIs with the heterogeneous crowd-sourced objects already present in the global map.

- **Practitioners' pain-point** – Municipal OGD stewards struggle to detect duplicates or stale entries when their official registers are imported into OSM. Inaccurate matches lead to downstream editing work and, in the worst case, vandalism accusations by the mapping community.
- **Scientific relevance** – Record linkage across heterogeneous spatial data is still an open research field [1, 2]. Existing approaches rarely consider both spatial and semantic similarity at scale for POIs.
- **Societal impact** – End-users of routing, tourism, and accessibility apps rely on the freshness of OSM. Poor conflation quality undermines public trust in open geodata.

The absence of a transparent conflation pipeline therefore affects *all* stakeholders—*OGD providers*, *OSM mappers*, *software developers*, and *researchers*. Subsequent chapters will show how the proposed solution bridges these needs.

1.2 Vision

> **Elevator pitch:** *"Within minutes, an OGD steward can publish a fresh POI dataset; the* `DiffedPlaces` *service automatically proposes high-precision matches and conflicts, ready for community review—thus keeping OSM both current and trustworthy."*

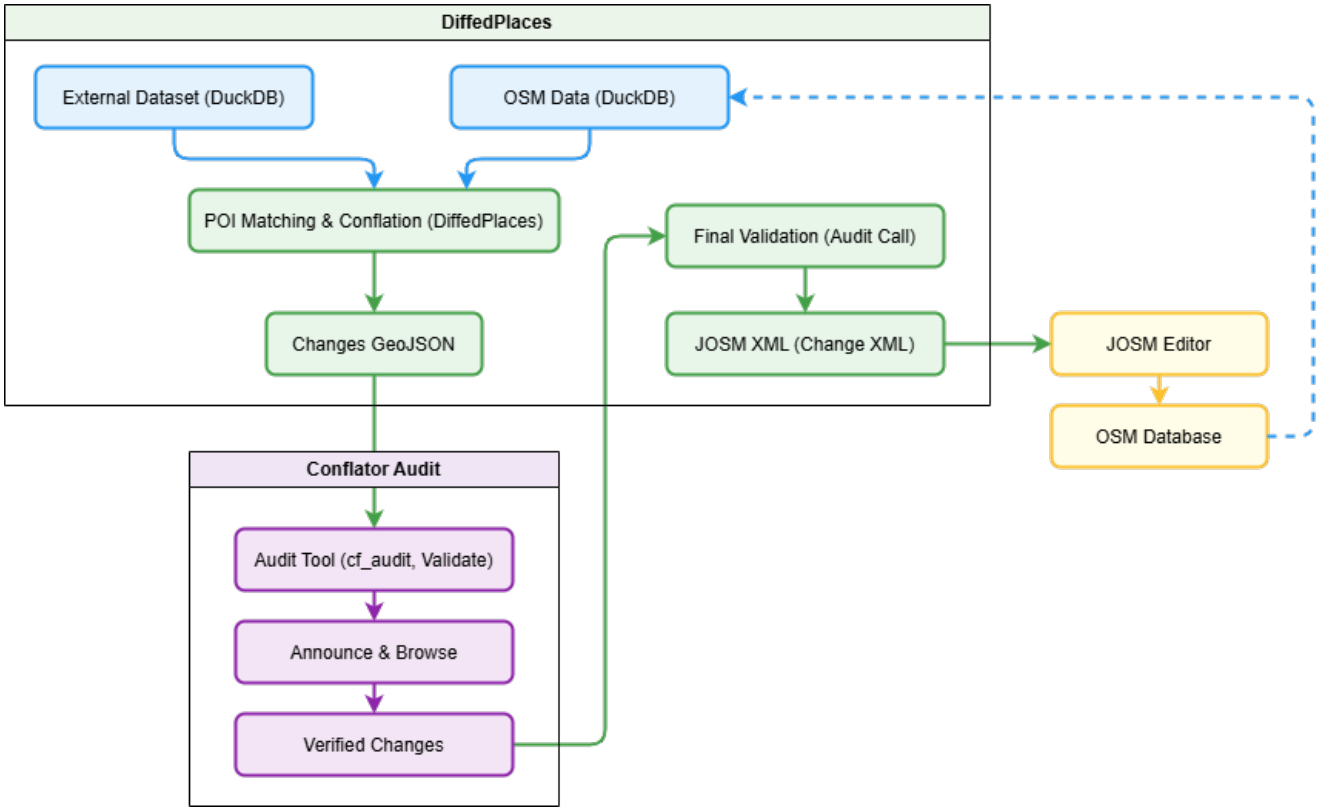

Figure 1.1: DiffedPlaces-centric workflow for POI conflation.

1.3 Objectives

The assignment "*Conflation of Point Data with OpenStreetMap*" (26 Jan 2025) specifies these objectives:

- O1: Extend the `DiffedPlaces` configuration to ingest Swiss ATP spiders for POIs.
- O2: Generate comprehensive diff outputs.
- O3: Implement a simple generic matcher based on heuristic measures.
- O4: Adapt and refine the PT1 algorithm ("Towards Automatic Points of Interest Matching", 2020).
- O5: Develop a standalone Python package that exposes matching logic and similarity scores.
- O6: Design evaluation metrics and validate on high-quality Swiss samples (dense vs. sparse regions; named vs. unnamed POIs).
- O7: Systematically compare simple and advanced approaches with emphasis on false-positive rates.
- O8: Integrate the best-performing algorithm into the production `DiffedPlaces` service and document its impact.

O9: Provide scientific insight into whether advanced methods are necessary for reliable Swiss POI conflation with OSM.

1.4 Boundaries and Constraints

Time & credits The thesis corresponds to 30 ECTS, carried out between **October 2024** and the submission deadline on **10 August 2025**.

Data scope Only point geometries are considered; linear or polygonal features (e.g. building footprints) are excluded.

Licensing All artifacts must remain compatible with the ODbL (for OSM).

Infrastructure The solution must run on commodity hardware ($\leqslant$ 16 GB RAM) and avoid paid cloud services.

Ethical aspects No personal data are processed; see Section 1.5 for details.

These limits focus the research on the core conflation challenge while ensuring that results can be adopted by the OSM community without legal frictions.

1.5 Data Ethics

The project processes only openly licensed, non-personal geospatial data from ATP spiders and OSM. No direct identifiers or sensitive attributes concerning individuals are handled. All inputs are subject to license compliance checks (notably ODbL) and are stored and processed on commodity hardware without external cloud services. Potential ethical risks (for example, inadvertent inclusion of personal data in upstream sources) are mitigated through conservative filtering, manual spot checks, and adherence to community import guidelines.

1.6 Methodological Approach

The work follows an *experiment-driven* paradigm centred on the provided `DiffedPlaces` tool:

1. **Literature review** – Analyse existing POI matching techniques and evaluation metrics (Chapter 2).
2. **Baseline experiment** – Run `DiffedPlaces` with deterministic rules on a Swiss test dataset.
3. **Error analysis** – Identify frequent false-positive/false-negative patterns; derive additional candidate features.
4. **Feature engineering & ML upgrade** – Extend the pipeline with Random Forest match; perform grid search and cross-validation.

5. **Field validation** – Deploy the improved pipeline on the golden data to compare it against the Rule-Based version.
6. **Field Test** – Use the approach to match a ATP spider and use the OSM Auditor to verify the matches.

1.7 Structure of the Thesis

- **Chapter 2** surveys the state of the art in spatial record linkage and outlines unresolved challenges for POI conflation.
- **Chapter 3** details the design and implementation of the `DiffedPlaces` pipeline, including evaluation results.
- **Chapter 4** documents the software-engineering aspects—architecture, codebase, CI/CD, and deployment.
- **Chapter 5** summarises contributions, discusses limitations, and suggests future research directions.
- **Appendices** provide glossary, dataset description, lab notebooks, and user-study protocol.

Chapter 2

Previous Work and Challenges

This chapter reviews previous efforts in OSM data matching and conflation, highlighting key tools and methodologies that have shaped the field. It also identifies the ongoing challenges in developing a semi-automated process for regularly updating and reconciling data, setting the stage for the improvements proposed in this project.

2.1 Related Work

The task of matching and conflating of POIs within geospatial data sources, such as OSM, is an increasingly essential area of research due to the proliferation of POI data from various sources. This section reviews key contributions to the field, focusing on methods that enhance POI matching, classification, and the handling of minimal metadata.

"Analyzing the Spatial-Semantic Interaction of Points of Interest in Volunteered Geographic Information" discuss the complexities inherent in Volunteered Geographic Information (VGI), particularly in the tagging and categorization of POIs within OSM. Their study emphasizes how community-driven labeling varies significantly, influenced by local conventions and contributors' interpretations, leading to challenges in consistent POI representation. This research highlights the need for tools that assist in maintaining the semantic and spatial integrity of POI data by analyzing spatial-semantic interactions and suggesting optimal tag recommendations. [3]

Research by "Classifying Points of Interest with Minimum Metadata" tackles POI classification under conditions of minimal metadata—namely, only names and coordinates. Traditional methods assume rich data with attributes like reviews and detailed descriptions. However, their work innovatively employs feature extraction based on surrounding spatial context and textual cues from names to achieve effective POI categorization. The approach demonstrates that with strategic use of OSM data and machine learning, accurate classifications are feasible, filling a notable gap in the literature. [4]

In the domain of POI matching, "Local POI Matching Based on KNN and LightGBM Method" introduce an advanced methodology combining KNN for local candidate set construction with re-encoded feature vectors using TF-IDF and BERT. Their binary classification model, built on LightGBM, outperforms conventional deep learning models like DNNs and SVMs, proving its efficacy in large-scale POI datasets where noise and redundancy are prevalent. [5]

The systematic review by "Conflating Point of Interest (POI) Data: A Systematic Review of Matching Methods" underscores the importance of POI conflation as a means to merge datasets that have partial overlaps or differing attribute focuses. This conflation process is crucial for improving data coverage and accuracy. They categorize existing POI matching methods and identify areas needing further exploration, particularly the integration of semantic and geometric attributes. [6]

"Mining the Co-existence of POIs in OpenStreetMap for Faulty Entry Detection" contributes by exploring the co-existence patterns of POIs within OSM to detect errors and enhance data reliability. Their work utilizes Tobler's first law of geography to examine spatial dependencies, thus aiding in identifying inconsistencies and suggesting corrections. [7]

These foundational studies lay the groundwork for continued advancements in POI data integration, emphasizing the necessity of robust algorithms capable of handling minimal data and ensuring the semantic consistency and spatial accuracy of matched POIs across diverse datasets.

2.2 Data Matching

The task POIs conflation begins by identifying pairs of records from two heterogeneous datasets (e.g. catalogs OGD and OSM) that refer to the * same * real world place. This subsection reviews core challenges and research strands that the literature has addressed.

Record linkage foundations. Early work on geospatial record linkage adapted generic duplicate detection techniques, such as the Fellegi-Sunter decision model with Winkler string comparators [2] and the error-tolerant indexing surveyed by Elmagarmid *et al.* [1]. These studies laid the algorithmic basis for later, spatially-aware matching schemes.

Spatial blocking and candidate generation. Because an $n \times m$ all-pairs comparison is infeasible for national-scale datasets, most pipelines first construct *candidate* pairs within a distance threshold ("blocking"). A recent systematic review shows thresholds ranging from $50\,\mathrm{m}$ to $1\,\mathrm{km}$, chosen ad hoc per use case [6]. Learning-based blocking has also emerged; for example, Xing *et al.* use a k-nearest-neighbour pre-filter before a LightGBM matcher, achieving $99.4\,\%$ accuracy on a $100\,000$+-pair benchmark [5].

Name and type similarity. POI names are short, noisy, and multilingual. Giannopoulos *et al.* exploit character n-grams and neighborhood context to classify under-specified POIs and improve type compatibility signals [4]. At the tag-scheme level, Mülligann *et al.* model the *spatial–semantic interaction* of OSM amenities, providing an empirical basis for semantic distance functions between categories [3].

Machine-learning matchers. Supervised models now outperform rule-based heuristics when there are sufficient labels. Piech *et al.* compared six learners and found random forests to be the most robust in five cities [8], while Low *et al.* demonstrated that Gradient Boosting edges out SVMs and rule sets in an end-to-end conflation framework [9]. Both studies confirm that combining textual, categorical, and spatial features is critical.

Quality control and validation. Manual inspection remains necessary for gold standard evaluation. Kashian *et al.* propose spatial coexistence rules to flag unlikely OSM entries, which can also surface false positives after matching [7]. In the transportation domain, Bast *et al.* quantify the performance of the classifier in the conflation of public transit stations, illustrating how domain-specific quality metrics drive algorithm choice [10].

Open challenges. Key gaps include (i) reusable and privacy-compliant benchmark datasets, (ii) multilingual name similarity that respects diacritics, (iii) uncertainty propagation from matching into downstream analytics, and (iv) annotation tooling that lowers the cost of producing labeled pairs, an essential prerequisite for state-of-the-art learning approaches.

In summary, data matching is the linchpin of POI conflation: robust candidate generation and multi-feature similarity modelling directly determine how much value later integration and enrichment steps can unlock.

2.3 Generalisation of External Data Processing

Motivation

Early integrations of OGD with OSM relied on bespoke *profiles* for the `osm_conflate` workflow.[11] While this strategy delivers highly-precise results for a single source, it does not scale to the tens of millions of POIs now published by projects such as ATP or by national open-data portals. Creating and maintaining a dedicated profile for every provider quickly becomes a bottleneck, as already observed during the PT1 similarity-matching study[12] and the follow-up PT2 prototype of the DiffedPlaces service.[13]

Data-Volume Challenges

- **Sheer size of the candidate space** – A naïve Cartesian comparison of two data sets with n and m points produces $n \times m$ pairs (complexity $\mathcal{O}(n\,m)$). Practical POI collections easily exceed $100\,000$ objects, turning the task into an intractable billion-pair problem. Blocking and k-nearest-neighbour pre-selection, as surveyed by Sun *et al.*,[6] reduce the workload by two to three orders of magnitude while preserving recall.
- **High-throughput reads and writes** – A single Switzerland-wide ATP spider already yields more than $75\,000$ candidate POIs; the complete ATP corpus reaches > 20 million rows. Classic PostGIS installations showed I/O saturation in PT2 load tests, prompting the switch to `QuackOSM`+DuckDB for columnar, in-memory analytics.[14]
- **Heterogeneous attribute schemas** – Street names, brands or wheelchair tags appear under different keys or value conventions. PT1 therefore introduced a mapping layer that normalises every source into a minimal, unified POI model before any comparison is carried out.[12]

Lessons Learned from PT1 & PT2

1. A *feature-vector* representation decouples matching logic from raw tags and enables generic ML classifiers such as the Random Forest used in PT1.[8]
2. Training data must cover the full variety of global OGD; otherwise the model over-fits to Swiss naming conventions. PT2 therefore automated dataset generation by sampling ATP spiders in batches of 200 – 25 000 POIs and measuring run-time and memory footprints.[13]
3. In duplicate-detection studies, blocking on coarse spatial buckets is known to cut the candidate set dramatically and often improves precision without hurting recall [1]

2.3.1 Centralized Data Integration Service

Building on these insights, PT2 proposed a *Centralized Data Integration Service* (CDIS) that orchestrates the full pipeline from raw web-scraped feeds to validated OSM upload files.[13]

Architecture. The CDIS is deployed as a Docker-compose stack with three core containers:

DB QuackOSM backed by DuckDB for fast ad-hoc SQL over billions of rows while keeping the footprint small enough for local development.[14]

Diff A worker that invokes `osm_conflate` followed by the PT1 classifier to generate human-readable *diff* files.[12]

API/UI A lightweight web front-end that exposes diff artifacts.[13]

Interoperability & Automation. Data import adapters convert GeoJSON, CSV or API payloads into the internal POI model at scheduled intervals (cron or message queue). A profile-free matcher is selected at run-time based on basic source metadata (e. g. country or typical distance error), eliminating manual configuration for new spiders.

Scalability Measures.

- Spatial proximity filtering partitions incoming POIs and OSM features so that only spatially proximate pairs enter the candidate pool, reducing PT2 diff processing time from hours to minutes for a whole country.
- Horizontal scaling is achieved by launching additional Diff workers; DuckDB's embedded nature avoids connection bottlenecks.
- All components are stateless except the DB; backups rely on immutable object storage snapshots, enabling rapid re-deployments.

Data-Quality Safeguards. Every match above a configurable confidence threshold is passed to an OSM Auditor instance for community vetting, keeping the human-in-the-loop principle established by the original `osm_conflate` workflow.[15]

Open Challenges.

1. **Random-Forest improvement** – The existing PT1 classifier needs deeper feature engineering (e. g. brand embeddings, hierarchical category encodings) and systematic hyper-parameter tuning to improve precision and recall.[13]
2. **Incremental re-training** – Human-validated *diffs* should be stored as labelled data so that the Random Forest can be refreshed regularly without full re-training whenever new ATP ground truth becomes available.
3. **Complex attribute support** – Extending the mapping layer to composite tags such as opening hours, wheelchair access and multipolygon sites is still open.
4. **Scalable national runs** – Country-wide conflation jobs continue to stress memory limits; spatial partitioning and chunked processing are required to meet the performance target in the assignment.

These observations confirm that a well-defined, centralized service is essential to keep pace with the ever-growing volume and diversity of external geospatial data while maintaining the quality standards expected by the OSM community.

2.4 Challenges in OSM Conflation

Conflation in the context of OSM refers to merging authoritative or community-curated external datasets with the existing volunteer-contributed map. Typical goals include enriching attributes, detecting omissions, and improving positional accuracy, while preserving the fine-grained provenance and licensing constraints that make OSM unique. A widely-used practical pipeline is the *OSM Conflator* script suite, which automates candidate generation, matching and change-file creation, and can be operated in a fully scripted or semi-manual review mode [15]. Within this thesis most low-level mechanics of conflation are therefore delegated to the `osm_conflate` logic; here we focus on the remaining research challenges and the relevant state of the art.

Matching quality vs. scalability Early rule-based workflows relied on tight spatial buffers and simple string metrics. Piech et al. demonstrated that even modest machine-learning models such as Random Forest already outperform hand-tuned thresholds once sufficiently labelled training data are available [8]. Nevertheless, these models still inherit the quadratic candidate-pair explosion and require careful blocking or KNN pre-selection, especially when global country-sized datasets are processed.

Semantic heterogeneity of categories The systematic review by Sun et al. highlights that POI classification mismatches (e.g. `amenity=restaurant` vs. `shop=fast_food`) are among the most frequent false-negative causes in conflation pipelines [6]. For this thesis we mitigate the issue by (i) adopting the high-level category mapping introduced in Section 3.4.1, and (ii) injecting those categories as hard constraints into the `osm_conflate` matching stage, thereby reducing both runtime and manual audit effort.

Data freshness and licence conflicts External OGD feeds can be updated daily, whereas the corresponding OSM geometry may lag behind or follow different update rhythms. The diff-based approach of `osm_conflate` preserves historic versions and lets the community review each proposed change set individually. Remaining licence compatibility checks (e.g. ODbL or local data-sharing agreements) are handled upstream in the data-ingestion pipeline and are therefore out of scope here.

Open challenges

- **Robustness to partial duplicates.** Chains of convenience-store branches often share identical names but differ in opening hours and brand tags. Extending `osm_conflate` with neighbourhood-based context features (nearby POIs, road-network density) could reduce such false positives.
- **Incremental re-training.** The machine-learning matcher should learn from each accepted or rejected suggestion. An on-device active-learning loop is preferable over cloud retraining to respect mapper privacy and network constraints.

- **Transparent feedback channels.** Tight interaction between data providers, the OSM community and public-sector stakeholders is required for long-term trust; surfacing conflation statistics via the DiffedPlaces dashboard is one concrete step in that direction.

2.4.1 Summary and Conclusion

This chapter synthesised the state of the art in conflating OGD with OSM. We first revisited classical record-linkage foundations—including probabilistic matching frameworks and string–similarity metrics—before surveying recent, data-driven approaches that combine spatial proximity with textual and semantic features to align POIs. Throughout, the practical limitations of heterogeneous schemas, divergent attribute completeness and naming conventions were highlighted, underscoring the need for robust preprocessing, normalisation and quality-assessment steps.

- **Baseline methods**: Deterministic and probabilistic techniques remain a reliable starting point for matching, but their effectiveness drops sharply when only minimal or noisy metadata are available.
- **Machine-learning advances**: Supervised models—especially ensemble learners trained on engineered spatial, lexical and contextual features—deliver markedly higher precision–recall performance when an adequately curated training set exists.
- **Spatial–semantic alignment**: Leveraging neighbourhood context and co-occurrence patterns improves disambiguation of dense urban POIs where geodesic distance alone is insufficient.
- **Workflow requirements**: A modular pipeline that integrates automated candidate generation, classification and human-in-the-loop verification is essential for safely importing external datasets such as ATP into OSM.

In sum, the chapter argues for a hybrid conflation strategy that marries geospatial computation with modern machine-learning techniques. These insights directly inform the design decisions of the *DiffedPlaces* system and the custom Random Forest Classifier–based matcher introduced in the next chapter, where we translate the conceptual findings summarised here into an operational toolchain for large-scale, high-quality data integration.

Chapter 3

Modern Approaches to POI Matching and Data Integration

This chapter lays out the methodology and experiments for conflating ATP with OSM: we define the case-study design (Section 3.1), present the end-to-end workflow (Section 3.2), then detail data sources and pre-processing (Section 3.3), candidate generation (Section 3.4), feature engineering (Section 3.5), and the matching approaches (Section 3.6). We validate and evaluate in Section 3.7, discuss limitations in Section 3.8, and close with a summary (Section 3.9).

3.1 Case-Study Design

This case study evaluates modern POIs matching techniques for integrating ATP data into OSM within Switzerland using the DiffedPlaces service. The goal is to assess a transparent rule-based matcher against a supervised alternative under controlled, reproducible conditions, and to deliver upload-ready artifacts for community review via OSM Conflator/osm_conflate and JOSM Editor.

3.1.1 Objectives & Scope

We aim to provide an evidence-based comparison between two POI matching paradigms within a well-defined geographic and operational boundary.

- **Comparative evaluation:** Benchmark a rule-based cascade against a supervised Random Forest Classifier using identical inputs and candidate sets.
- **Operational relevance:** Quantify performance under typical Swiss urban and rural settings, including cases with sparse or noisy metadata.

- **Actionable outputs:** Produce consistent diffs and upload-ready ChangeXML that can be audited and, where appropriate, integrated into OSM.

Scope. The study is restricted to Switzerland to ensure geographic consistency and to leverage high-quality local mapping. External inputs originate from ATP; the OSM baseline is a recent country extract or planet cut filtered to relevant POI tags. Licensing constraints are respected under ODbL [16].

3.1.2 Experimental Design

The experiment is structured to enable fair, repeatable comparison and ablation.

- **Dataset selection & preparation.** We filter several ATP spiders to Switzerland and extract OSM nodes/ways with *amenity*, *shop*, *tourism*, *leisure*, and *office* tags. Records are normalised (names, brands, categories) and staged in DuckDB for fast, columnar scans.
- **Candidate generation.** Spatial proximity filtering (distance/radius queries) combined with coarse category compatibility reduces pair explosion; details in Section 3.4.
- **Algorithms under test.**
  1. *Rule-based:* distance threshold, category compatibility, and name similarity with deterministic tie-breakers.
  2. *ML-Based Matcher:* Random Forest Classifier over lexical, spatial, semantic, and data-quality signals [8, 9].
- **Validation framework.** Precision, recall, and F1-Score, plus runtime and throughput; stratified analysis for urban/rural subsets.
- **Ablations & thresholds.** Lexical-only vs. lexical+spatial vs. full features, and threshold sweeps for both matchers to inspect precision–recall trade-offs.

3.1.3 Design Considerations

- **Geographic focus.** A single-country scope simplifies coordinate sanity checks and reduces heterogeneity in address and category conventions.
- **Semantic prefiltering.** High-level categories reduce candidate pairs and improve disambiguation (cf. Section 3.4).
- **Efficiency & reproducibility.** DuckDB enables repeatable, fast preparation; pipelines are idempotent to support re-runs when upstream or downstream drift occurs (cf. Section 3.2).
- **Human-in-the-loop.** Outputs are intentionally channelled through auditing tools (e.g., cf_audit) and final review in JOSM Editor to preserve mapper oversight.

- **Licensing & ethics.** Compatibility with ODbL is enforced; we avoid importing unverifiable or copyrighted content and document provenance for each external source [16].

3.1.4 Expected Outcomes

- A documented end-to-end workflow from ingestion to upload-ready ChangeXML, with reproducible settings.
- A comparative analysis quantifying strengths/weaknesses of rule-based versus ML matching on Swiss POIs.
- Practical guidance on category design, blocking, and thresholding for geographically constrained conflation.

3.1.5 Threats to Validity

Internal validity may be affected by labelling errors in ground truth and by covariate shift between cantons; external validity is limited to datasets similar to ATP and to regions with comparable mapping density. We mitigate these risks via stratified evaluation, ablations, and by publishing configuration and data slices for replication.

3.2 Overview of POI Matching Workflow

This section explains the final matching workflow with the DiffedPlaces service included in detail. Therefore, this secion walks through every element of Figure 1.1 in the order indicated by the arrows and explicitly references the colour scheme used in the diagram:

(1) Database Ingestion – blue boxes Fresh extracts of the external OGD feed and the current OSM planet file are imported into a local `DuckDB` instance. This step is deliberately stateless: whenever newer upstream data become available, the importer simply truncates and reloads the staging tables, thereby restarting the entire pipeline without residual side-effects.

(2) POI Matching & Conflation – green boxes Inside the **DiffedPlaces** service, an enhanced version of `osm_conflate` resolves one–to–one, one–to–many and many–to–one correspondences between candidate POIs. The matcher can be swapped (cf. rule-based vs. ML-based engines in Section 3.6) without touching downstream artifacts because every variant emits a unified *Changes GeoJSON*. Geometry repair, tag merging and conflict resolution are already carried out at this stage so that subsequent auditing focuses on semantic correctness rather than syntactic clean-up.

(3) Crowdsourced Validation – purple boxes The *Conflator Audit* module encapsulates three progressively stricter review layers:
1. the command-line `cf_audit` helper and its optional GUI wrapper `validate`,

2. a web dashboard for *announce & browse* that encourages local mappers to comment on or veto the diff
3. a *verified changes* queue that must reach a quorum before edits graduate to the next step.

(4) Re-validation & ChangeXML Generation – green boxes Verified patches are re-ingested into the matcher for a final consistency check. Only if the audit diff is still *clean*—i. e. no conflicting edits were introduced on the live OSM database in the meantime—does the pipeline emit a standards-compliant JOSM *Change XML*. This additional safeguard protects against accidental reversion of high-quality community contributions.

(5) Human-in-the-loop Upload – yellow boxes Finally, the generated ChangeXML is opened in the desktop JOSM editor, allowing a last visual sanity check and optional manual tweaks (e. g. conflating neighbouring POIs or aligning building outlines). When the uploader commits the changeset, the official OSM database is updated and, after the normal planet-file export cycle, feeds back into step (1), closing the loop indicated by the dashed blue arrow.

Iterative Operation. Because every stage is idempotent and outputs clearly versioned artifacts, operators can schedule the entire workflow as a daily or weekly cron job. In practice, two scenarios trigger a new run:

- *Up-stream drift* – the authoritative ATP provider publishes an updated stop list or timetable, causing the external dataset to diverge from OSM.
- *Down-stream edits* – large community mapping parties or automated editors significantly reshape a region, invalidating previous match decisions.

Both cases can lead to a restart of the pipeline with fresh blue input blocks. Consequently, the proposed architecture achieves continuous integration of heterogeneous POI sources while preserving the essential human oversight embodied in the purple audit stage.

3.3 Data Sources & Pre-processing

Source description. The pipeline combines retailer and brand feeds from ATP [17] with an OSM baseline. For ATP, spiders focused on Switzerland provide outlet-level POIs with names, categories, addresses, and coordinates. For OSM, a recent extract (planet cut or country PBF) offers community-curated features and tags that reflect on-the-ground reality under the ODbL license [16]. All payloads arrive as GeoJSON, CSV, or API responses and are staged in DuckDB for reproducible, fast scans [14]. A national boundary mask ensures that only features within Switzerland are considered.

Attribute normalisation. ATP already provides OSM-like tagging. Therefore, normalization was not a focus.

Coordinate cleaning / CRS. All geometries are unified in WGS84 (EPSG:4326). Basic sanity checks flag out-of-bounds coordinates and potential lat/lon swaps, and implausible points are quarantined for later review rather than silently dropped.

Filtering. The OSM baseline is reduced to features likely to represent POIs (for example, amenity, shop, tourism, leisure, or office). On the ATP side, known spider-specific noise is removed using conservative inclusion rules that prefer precision over recall to avoid false positives early in the process.

ATP spider catalogue. The ATP corpus draws from a broad catalogue of brand and institution spiders with visible presence in Switzerland. It covers major grocery and retail chains (for example, Denner, Migros, Lidl, Volg, FUST, Jumbo), food and beverage (Five Guys, Dunkin', L'osteria), health and beauty (Coop Vitality, Fielmann, Marionnaud, Misenso), mobility and energy (IONITY, Tesla, "Ich tanke Strom"), accommodation (Accor, Marriott, Ritz-Carlton, Wyndham), fashion and lifestyle (Adidas, H&M, LEGO, Pandora, TAG Heuer), as well as public services and municipalities (City of Zurich, Winterthur, emergency meeting points). This breadth ensures coverage across everyday, high-street, and travel-related POIs. A small sample spider is retained for smoke checks and end-to-end dry runs. The full list of spider identifiers is provided in Appendix E.

Category system & tag harmonisation. To make heterogeneous sources comparable, both ATP categories and OSM tags are harmonised into a concise set of high-level classes (for example, retail, gastro, health, transport, leisure, finance, public, service, education, accommodation, and other). OSM tags such as amenity, shop, tourism, office, or healthcare are mapped into this taxonomy so that like is compared with like during conflation. This abstraction reduces incidental differences in naming and tagging into consistent signals for subsequent matching and evaluation.

Temporal snapshots & provenance. Both ATP and OSM inputs are handled as dated snapshots. Each snapshot carries a source identifier, retrieval date, and, where available, record-level update timestamps. This provenance trail supports reproducibility and helps interpret apparent discrepancies that arise from asynchronous updates across sources [12, 13].

Address handling. Addresses exhibit local spelling conventions and abbreviations. A lightweight normalisation expands common abbreviations, aligns street types, and standardises postal codes and locality names for Swiss data. The intent is not to overfit to one schema, but to make obviously equivalent addresses comparable while preserving original strings for audit.

Spatial scoping. A country boundary mask focuses the analysis on Switzerland.

Golden data for calibration. A small, curated "golden" subset of retailer outlets (for example, a single grocery brand) is maintained to calibrate thresholds and to qualitatively validate the end-to-end conflation. This dataset anchors expectations for category mapping, address normalisation, and positional tolerances without biasing the broader evaluation.

Provide exact snapshot date ranges and OSM extract source; and the boundary dataset version. See Appendix **??** for the spider identifiers.

3.4 Candidate Generation (Blocking)

The goal of blocking is to keep the subsequent matching tractable while maintaining high recall. Instead of attempting all $O(n \cdot m)$ pairings, the method constructs a small pool of plausible pairs using simple spatial cues and light semantics, deferring fine distinctions to later stages.

Spatial blocking. A proximity screen selects OSM features within a plausible neighbourhood of each ATP outlet. The neighbourhood size reflects typical geocoding uncertainty, store footprint, and urban context. A national boundary mask (Switzerland) prevents spurious cross-border candidates.

Semantic prefilter. Only records that agree on a high-level category are considered as candidates (for example, retail with retail, health with health). This prefilter relies on the shared taxonomy described in Section 3.4.1 and Appendix E. It reduces accidental pairings across unrelated domains while preserving variety within a category.

Temporal consistency. Candidate generation considers contemporaneous snapshots so that openings, closures, or relocations do not dominate the pool. Snapshot identifiers and retrieval dates recorded in the data provenance help interpret apparent mismatches that are time-related rather than spatial or semantic.

This approach reflects common practice in record linkage and POI conflation [6].

3.4.1 Derivation and Iterative Validation of High-Level Categories

The high-level semantic categories ("superclasses") used for blocking were not adopted wholesale from any external taxonomy; they were iteratively derived and validated using internal analysis utilities ("devtools") to ensure (i) coverage across diverse POIs supplied by ATP, (ii) discrimination power (preventing implausible pairings), and (iii) stability for feature engineering and model training.

Tooling. Scripts in the internal `devtools` directory (e.g., `analyze_atp_spiders.py`, `analyze_atp_categor` `analyze_osm_category_mapping.py`) were executed over successive dated ATP output snapshots to:

- Summarise, per spider, the distribution of assigned superclasses.
- Surface counts still falling into the fallback category `other` (an indicator of mapping gaps).
- Enumerate missing or newly observed underlying OSM-like keys / values for consideration in the mapping.
- Produce GeoJSON extracts (e.g., `atp_other_categories.geojson`) of unresolved or ambiguous items for manual inspection.

Initial state (Run 1). Early runs showed substantial leakage into the generic `other` superclass for certain spiders. Examples (Run 1 log excerpts): `avis` (4'818 POIs) and `sixt` (2'082 POIs) were entirely classified as `other`; `discover_swiss` produced a mixture with a residual `other` count (log excerpt indicates non-accommodation remainder); municipal / public-service sources (e.g., `winterthur_ch`) also exhibited large `other` counts (2'016 of 2'224 POIs).

Mapping gap analysis. The JSON report `atp_missing_main_keys.json` highlighted amenity / tourism / office values absent from the enumerated mapping (e.g., `bench`, `bicycle_rental`, `casino`, `charging_static` `recycling`, `toilets`, `vending_machine`, `tourism:apartment`, `office:consulting`, `office:estate_ag` `office:it`). Inclusion or explicit handling of these values reduced misclassification into `other` and informed the categorical feature list (Appendix D).

Refinement (Runs 2/3). After updating the mapping rules:

- `avis` and `sixt` reclassified from `other` to `transport` (4'818 and 2'082 POIs respectively).
- `discover_swiss` consolidated under `accommodation` (3'957 POIs) once lodging-related tags were normalised.
- `winterthur_ch` shifted a large share from `other` to a dedicated `public` superclass (2'016 POIs in Run 2/3 excerpts).
- `our_airports`: previously classified predominantly as `other`, re-mapped to `transport` (70'450+ POIs in a later run excerpt).

Decision criteria. A candidate subclass is promoted to a superclass only if it meets all of: (i) distinct semantic role affecting downstream matching plausibility (e.g., separating `transport` from generic retail reduces false positives between fuel stations and shops), (ii) sufficient volume to justify a blocking

partition, and (iii) stable tagging patterns across both ATP and OSM. Otherwise, values are merged into an existing superclass or retained under `other` pending further evidence.

Outcome. Iterative refinement reduced the share of POIs falling into `other` and increased alignment between brand-specific spiders and their expected semantic classes (e.g., car rental to `transport`). This improved blocking precision without materially harming recall.

Feedback loop. Each refinement cycle (1) should run the analysis scripts, (2) inspect large `other` contributors, (3) update mapping tables / normalization rules, (4) regenerate statistics, and (5) freeze a versioned mapping for reproducibility across experiments. Deviations (e.g., sudden growth in `other`) act as drift signals prompting reassessment of upstream spider outputs or newly emerging tags.

The process ensures that category-based blocking reflects real semantic compatibility rather than brittle string matches, forming a reliable first-stage filter ahead of similarity scoring and learning-based classification.

3.5 Feature Engineering

This section outlines the goals, scope, and methodology for crafting features used to decide whether an POI from ATP corresponds to a feature in OSM. The emphasis is on the ideas and process rather than implementation specifics; detailed rule heuristics and model-specific signals follow in Section 3.6 and the validation in Section 3.7.

Goals and scope. The feature set is designed to be: (i) robust to noisy inputs (brand aliasing, spelling variants, partial addresses), (ii) conservative with respect to long-range spatial pairings, (iii) language-aware for the Swiss context, and (iv) extensible via configuration without code changes. Features operate on candidate pairs after spatial blocking (Section 3.4) and before the matching approaches (Section 3.6).

Methodology. Feature definition followed an iterative, analysis-driven process: (1) inspect data distributions and frequent error modes, (2) propose simple, explainable signals targeting those errors, (3) sanity-check their behaviour on curated samples, and (4) retain only signals that generalise across brands, categories, and regions. Internal analysis utilities supported this loop by reporting coverage, outliers, and category balance, and by highlighting where signals disagreed—useful for diagnosing false positives and false negatives early.

Data harmonisation first. Before any pairwise signal is computed, inputs are cleaned and canonicalised in a non-destructive way: normalised casing and Unicode, punctuation trimming, and token-level simplifications; duplicate whitespace collapsing; and safe fallbacks for missing or malformed fields. Harmonisa-

tion intentionally stays lightweight to avoid hiding real inconsistencies that are valuable for downstream decisions.

Language and naming considerations. Given Switzerland's multilingual setting, name features aim to be tolerant to diacritics, common abbreviations, and interchangeable descriptors (for example, "Bahnhof"/"Gare"). Brand/operator fields, when present, are treated as complementary signals to free-text names. Stopword lists and token-set style comparisons prioritise content words over generic terms. The objective is to recognise the same outlet written in slightly different ways rather than to perfect linguistic normalisation.

Addresses and spatial context. Address elements are used as weak consistency checks (street agreement, house-number proximity) rather than hard gates due to known incompleteness on either side. Spatial proximity is transformed into monotonic indicators (for example, within distance bands) to stabilise decisions across urban and rural settings. Where neighbourhoods are dense (stations, malls), the design prefers features that can down-rank near-duplicates without discarding plausible candidates outright.

Analysis tooling and configuration. Throughout development, lightweight analysis scripts were used to quantify category coverage, study excluded features, and visualise disagreements between signals. Together with a configuration-driven category mapping and thresholds, this made the feature layer auditable and easy to adjust as the dataset evolved, without changing the core code.

Similarity assessment methods. Similarity is assessed through complementary lenses rather than a single metric: token-based and character-based name similarities (for example, token-set and Jaro–Winkler) emphasise robustness to ordering and minor edits [2, 18]; vector-space comparisons (character n-gram TF-IDF cosine) capture broader lexical overlap including partial names; address agreement treats street and house number as supportive cues with fuzzy matching to accommodate abbreviations; spatial proximity is translated into bands or smooth decays to stabilise decisions across densities; and semantic agreement (high-level category compatibility) filters implausible pairs early [10]. These views can be combined by transparent rules or a supervised matcher (Section 3.6), with calibration and validation deferred to Section 3.7.

We compute features for each candidate pair, motivating each signal by its contribution to disambiguation:

Lexical [2, 18]. Normalised name similarity (token-set ratio), Jaro–Winkler, and character n-gram TF-IDF cosine. Brand/operator equality; common prefix/suffix flags.

Spatial. Haversine distance; within-threshold booleans (e.g., 50 m, 100 m). Same-address heuristic (if address data present).

Semantic. High-level category equality and compatibility; optional neighbourhood context density.

Data-quality flags. Missing house number; missing name; age of source record.

Feature catalogue and normalisation

The final catalogue consolidates the signals above and standardises their ranges for consistent downstream use:

- Name token-set ratio in $[0, 1]$ (scaled from a $[0, 100]$ score).
- Jaro–Winkler similarity in $[0, 1]$.
- Character n-gram TF-IDF cosine in $[0, 1]$.
- Brand/operator equality as binary flags $\{0, 1\}$; shared prefix/suffix overlap ratio in $[0, 1]$.
- Street-name agreement in $[0, 1]$ (fuzzy similarity on normalised street strings)
- house-number proximity encoded as distance bands (same, ± 1, ± 2+) as $\{0, 1\}$ indicators.
- Haversine distance d (metres) represented as: (i) band indicators for $\leqslant 25$, $\leqslant 50$, $\leqslant 100$, $\leqslant 250$, $\leqslant 500$, $\leqslant 1'000$, $\leqslant 1'500$ m; and (ii) an optional monotonic decay score $s = \exp(-d/\tau) \in [0, 1]$ for smoothness.

Normalisation policy: all similarity measures are constrained to $[0, 1]$; boolean cues are $\{0, 1\}$. Spatial distances are not z-scored; they are expressed as bands and/or a bounded decay to avoid dependence on dataset-specific scales. When fitted statistics are needed (for example, vocabulary or IDF weights), they are derived on training folds only to prevent leakage.

Importance assessment. Per-feature contribution is assessed without committing to a single estimator: (i) embedded importances from tree-based models (mean decrease in impurity), (ii) permutation importance on held-out folds for model-agnostic sensitivity, and (iii) stability checks across cross-validation folds.

In qualitative terms, short-range spatial bands and name similarities are consistently the strongest disambiguators; high-level category agreement prevents off-type pairings; address and quality flags act as reliable tie-breakers when present. Detailed, model-specific figures are reported in Section 3.7.

3.6 Matching Approaches

This section details the alternative strategies used to decide whether an POI from ATP corresponds to an OSM element after spatial blocking (Section 3.4) and feature engineering (Section 3.5). Two complementary paradigms are implemented: (i) a deterministic, interpretable rule-based matcher that com-

poses weighted similarity components, and (ii) a supervised Random Forest Classifier-based matcher that consumes a feature vector (partly reusing the same underlying similarity computations) to predict match probability. A hybrid decision policy then leverages the strengths of both. Validation procedures and quantitative evaluation are intentionally deferred to Section 3.7.

3.6.1 Rule-Based Matcher

The rule-based matcher assigns a composite similarity score to each candidate pair and classifies it as a match if the score exceeds a configurable threshold. Its design goals are (i) transparency (each component score and its weight is logged), (ii) robustness to noisy or incomplete tags, and (iii) predictable behaviour under configuration changes.

Configuration profiles. Three predefined configurations balance precision/recall trade-offs were used for assessment:

- **Standard**: balanced weights across geometric (0.15), name (0.20), brand (0.13), operator (0.05), address (0.08), opening hours (0.05), phone (0.08), website (0.08), and generic tag overlap (0.13); similarity threshold 0.85.
- **Geometry-focused**: increases geometric weight (0.30) to favour very close spatial pairs when textual tags are sparse; threshold 0.85.
- **Geometry+Name focused**: emphasises geometry (0.30) and name (0.30) while dropping weak or noisy components (for example, opening hours, website) by setting their weights to `None`; raised threshold 0.875 for higher precision in dense areas.

Components with a missing or `None` weight are excluded from the aggregation (their similarity may still be computed for diagnostics). A maximum matching distance (global configuration) acts as a hard spatial gate: if the Haversine distance exceeds this value the pair is immediately rejected with reason metadata.

Similarity components. For eligible pairs (within distance):

Geom A monotonic decay based on distance, lower-bounded at 0.5 for any pair within the maximum distance to avoid collapsing scores for otherwise strongly matching attributes. This prevents very small positional noise from dominating.

Name Primary name similarity, with fallbacks to brand/operator cross-comparisons. Multiple candidate similarities are computed and the maximum retained. This captures cases where a dataset encodes a brand in `name` while OSM stores it under `brand` (or vice versa).

Brand / Operator Direct string similarity if both sides provide the tag; otherwise the component yields *None* (excluded) rather than zero to avoid penalising missing-but-valuable attributes.

Address Constructed by concatenating relevant address tags (street, house number, postcode, city, country, fallbacks) into canonicalised strings before fuzzy comparison. Address similarity is intentionally down-weighted because of incompleteness and heterogeneous formatting.

Opening hours / Phone / Website Normalised plain-text comparisons after simple sanitisation (for example, stripping formatting characters in phone numbers). Each acts as a strong tie-breaker when present and equal, but does not block matches when absent.

Tag overlap Jaccard-style similarity over the set of remaining informative tags after excluding globally ignored keys (licence, metadata, volatile counters) and address/contact keys. This measures semantic consistency beyond the explicitly weighted fields.

String comparison methods. Each component uses a method selected in the active configuration: Jaro–Winkler, character-level Jaccard, Levenshtein ratio, or token-sort ratio (order-insensitive fuzzy matching). Methods can be disabled per component (none), in which case that component returns *None*. All methods output a value in $[0, 1]$.

Name fallback logic. The matcher collects several similarities: (name, name), (name, brand), (brand, name), (name, operator), (operator, name). Empty strings are skipped. The best available similarity is used as the *name* component, increasing recall where branding choices differ between sources.

Weighting and thresholding. Let C be the set of components with both a numeric score and a defined weight; the final score is $\min(1, \max(0, \sum_{c \in C} w_c s_c))$ after normalising implicitly by using only the present weights (i.e., weights of missing components are not redistributed). A pair is classified as a match if the score exceeds the configuration's similarity threshold. This design avoids abrupt jumps when optional attributes (e.g., website) appear, while ensuring their presence still increases confidence proportionally.

Auditing and interpretability. For every evaluated pair the matcher records: raw distance, per-component scores, chosen methods, applied weights, final score, threshold, and (if rejected) the reason (distance gate, lack of sufficient similarity, or competition with a higher-scoring candidate). This metadata underpins subsequent error analysis and facilitates reproducible tuning of per-category overrides.

Deterministic candidate resolution. When multiple OSM candidates compete for one source record within the spatial threshold, the highest final score wins. Ties are resolved deterministically by (i) shorter distance, (ii) brand/operator exact equality presence count, then (iii) higher raw name similarity before fallback. Remaining candidates are flagged for manual review rather than auto-rejected to reduce false negatives.

3.6.2 ML-Based Matcher

The ML matcher formulates candidate resolution as supervised binary classification with a Random Forest Classifier. It consumes a deterministic feature vector whose primitives reuse the exact similarity functions of the rule-based matcher (geometry decay, string similarities, address assembly, tag filtering). This deliberate reuse removes training/serving skew and ensures that differences in behaviour stem from learned non-linear combinations instead of divergent preprocessing.

Processing pipeline. For each dataset POI: (1) spatial blocking proposes nearby OSM candidates (Section 3.4); (2) the feature calculator derives the ordered feature vector; (3) the pickled forest model (reloaded at the start of each diff run) outputs a match probability; (4) a probability threshold selects at most one winning candidate (ties: higher probability, then shorter distance, then higher primary name similarity); (5) the match is registered identically to other strategies.

Exact feature ordering (current implementation). The implemented feature vector is intentionally lean and its ordering is stable. The first seven features are exactly the rule-based component scores defined in Section 3.6.1. Subsequent features add compact structural and semantic signals (categorical encoding details are fully documented in Appendix D, referenced instead of repeating exhaustive lists here):

1–7) Rule-based components (refer Section 3.6.1): `geom`, `name`, `brand`, `operator`, `address`, `phone`, `tag_overlap`.

8) `source_has_name` (1 if dataset POI has non-empty name, else 0).

9) `osm_has_name` (1 if candidate OSM element has non-empty name, else 0).

10) `name_length_ratio` (shorter name length divided by longer; neutral 1 if both empty, 0 if only one empty).

11) `tag_count_ratio` (min(tag count) / max(tag count); neutral 1 if both zero, 0 if only one zero).

12–24) Tri-state main key features: `tag_main_<key>` for each top-level configured key sorted alphabetically (current set e.g., `amenity`, `aerialway`, `aeroway`, `building`, `craft`, `emergency`, `healthcare`, `highway`, `leisure`, `office`, `shop`, `tourism`). Value meaning: 0 neither side has the key; 1 only one side has it; 2 both sides have it (key present on both regardless of differing values).

24–147) Tri-state specific value features: `tag_specific_<key>:<value>` for every enumerated value in the configuration (e.g., `amenity:restaurant`, `amenity:cafe`, representative `shop:*` values, etc.), sorted alphabetically as a single list. Value semantics: 0 neither side has that exact key:value; 1 only one side has it (asymmetric); 2 both sides share the exact key:value (strong semantic agreement).

The tri-state (0/1/2) design compresses presence, asymmetry, and agreement for both general keys and specific categories into one numeric feature each, avoiding proliferation of separate one-hot columns per side while retaining discriminative power. Persisted models rely on this exact ordering; changes require retraining.

Relationship to conceptual feature set. Earlier design options (distance band one-hots, per-field address similarities, website/ opening hours similarities, interaction terms) are presently *not* part of the production vector. They remain future extensions. The current approach reuses only the core rule-based component scores plus compact semantic coverage through tri-state tag encodings.

Geometry representation. The only geometry-derived feature is the rule-based `geom` component (a distance decay with lower bound). Raw distance and band indicators are intentionally omitted to reduce correlation and dimensionality.

Name / brand / operator / address signals. These appear solely via their aggregated rule-based component scores and the two name presence flags plus a length ratio. Fine-grained cross-similarity variants are reduced to a single composite value (the rule-based component already applies internal fallback logic). This keeps overfitting risk low when labelled data is sparse.

Tri-state semantic encoding. Each main key and each specific key:value mapping yields a single ternary feature (0 absent on both, 1 asymmetric, 2 agreement). This compresses presence, asymmetry, and agreement; the full catalogue of keys/values and rationale is provided in Appendix D.

Derived ratios. `name_length_ratio` and `tag_count_ratio` normalise structural differences. Both are symmetric ($\in [0, 1]$) and return 1 when both sides are empty (neutral) and 0 when only one side is empty, distinguishing absence from mismatch.

Encoding rules. All component scores are already in $[0, 1]$. Tri-state semantic features use the closed set {0,1,2}. No additional presence flags are required because 0 vs. 1 vs. 2 disambiguates absence, asymmetry, and agreement directly. Missing names affect the name presence flags and indirectly influence downstream splits.

Handling missing data. Absence patterns are informative: (0 distance band “very close” + missing brand on both sides + strong name similarity) might still yield a match, whereas (moderate distance + generic name + asymmetric category presence) likely does not. Presence flags thus allow the forest to attach neutral or negative weight according to empirical correlations rather than a priori assumptions.

Why a Random Forest. The forest copes well with heterogeneous numeric and categorical mixtures without the need for scaling, offers competitive accuracy for tabular data, exposes feature importance for iterative pruning, and keeps inference latency negligible for batch diff processing [**?**, 19]. More complex ensembles (e.g., gradient boosting) or representation learning approaches were deferred until diminishing returns of the current setup justify added complexity.

Scope. Empirical calibration, threshold tuning, and performance curves are reported separately (Section 3.7).

3.6.3 Training of Matchers

This subsection summarises how the two matcher families are tuned and trained. Detailed quantitative results are deferred.

Rule-based parameter search. The rule-based matcher exposes weights and per-component string similarity methods. A search resembling grid search over a constrained space of weight vectors and method choices was executed against high-quality golden data. Golden data must be both precise (low labelling noise) and exact in spatial / attribute alignment; otherwise weight optimisation overfits artefacts. Candidate configurations (e.g., Standard, Geometry-focused, Geometry+Name) were evaluated on precision / recall trade-offs, and the best-performing sets retained as named profiles. Components prone to noise in dense urban contexts (e.g., website, opening hours) were selectively disabled (weight `None`) in the high-precision profile. Thresholds were then tuned secondarily; distance gating remained fixed by global configuration.

ML training phases. The ML matcher undergoes staged training:

1. **Synthetic bootstrap**: An initial training corpus is generated programmatically (artificial positive pairs via mild perturbations of genuine POIs; artificial negatives via mismatched category swaps, spatially distant pairs, and near-miss conflicts). This furnishes balanced, diverse examples without manual labelling overhead and allows early feature sanity checks.
2. **Incorporation of deterministic labels**: The calibrated rule-based matcher is run over real candidate sets. Very high-scoring pairs (above a conservative precision-oriented threshold) are appended as additional positives; very low-scoring pairs (below a low threshold) become negatives. A margin band in between is excluded to avoid propagating ambiguous labels. When multiple matches are possible for the same entity, only the highest-scoring pair is retained as a positive match; all other pairs, regardless of score, are treated as non-matches. This assumption reflects the expectation that true duplicates in OSM are rare, and it enriches the training data with hard negatives—pairs that may appear similar but are in fact distinct. This injection introduces authentic noise patterns absent from purely synthetic generation and measurably improves generalisation.

3. **Mixed retraining**: A train/test split (configurable fraction, stratified) is applied. Hyperparameters (tree count, depth, minimum samples split / leaf, feature subsampling strategy) follow a defined parameter set; a grid-like exploration defined in configuration selects a performant combination (e.g., via accuracy or balanced metrics on validation). The chosen model is persisted (pickle) with metadata (hyperparameters, timestamp) for reproducibility.

Continuous improvement. Each diff processing cycle reloads the latest persisted model to pick up improvements without redeploying code. Additional future augmentation sources (e.g., manually audited discrepancies, active learning queries) can be appended following the same high-confidence label gating principle.

Complementary roles. The rule-based search ensures an interpretable baseline and supplies high-confidence pseudo-labels; the ML model captures interaction effects and smooths decision boundaries. Together they reduce manual review volume while maintaining precision.

3.6.4 Final Operational Parameters

This subsection records the concrete deployed ("final") parameters for (i) the rule-based matcher under its high-precision *Geometry+Name focused* profile (Section 3.6.1) and (ii) the Phase 2 augmented Random Forest Classifier (Section 3.6.2). These settings generated the evaluation results discussed in Section 3.7. Listing them here supports auditability and reproducibility.

Rule-based matcher (Geometry+Name focused). Emphasises geometry and primary name similarity; disables sparse / noisy components. Only non-null weights contribute (no re-normalisation). A distance gate rejects pairs beyond the maximum distance before similarity aggregation.

ML matcher (Random Forest). Hyperparameters of the persisted Phase 2 model. Feature ordering / semantics as per Section 3.6.2. A probability threshold (0.50) selects at most one winning candidate per source POI; post-calibration analysis indicated this operating point provided a balanced precision/recall trade-off without further adjustment.

Any future change (e.g., feature pruning, calibration, distillation) should update this subsection with versioned metadata and justification.

3.6.5 DB Service CLI Interface

A small internal CLI complements the HTTP API for quick admin and developer tasks without starting the full stack. It reuses the same code paths as the service, favors idempotent operations, and is meant for setup, diagnostics, and fast iteration.

Table 3.1: Final rule-based matcher configuration.

Parameter	Value	Notes
Maximum distance	1'500 m	Hard spatial gate (fixed)
Similarity threshold	0.875	Precision-leaning operating point
Weight (geom)	0.3000	Distance decay score (lower bound applied)
Weight (name)	0.3000	Composite name similarity (fallback logic)
Weight (brand)	0.1000	Brand similarity if present
Weight (operator)	0.0500	Operator similarity if present
Weight (address)	0.1000	Canonicalised address similarity
Weight (phone)	0.0500	Normalised phone similarity
Weight (tag_overlap)	0.1000	Jaccard-style informative tag overlap
Weight (opening_hours)	None	Disabled (sparse / noisy)
Weight (website)	None	Disabled (sparse / noisy)
Method (fuzzy_name)	jaro_winkler	Slight recall lift; higher global threshold
Method (fuzzy_brand)	token_sort_ratio	Order-insensitive
Method (fuzzy_operator)	token_sort_ratio	Order-insensitive
Method (fuzzy_address)	token_sort_ratio	Robust to token reordering
Method (fuzzy_phone)	levenshtein	After digit normalisation
Method (fuzzy_oh)	None	Disabled component
Method (fuzzy_website)	None	Disabled component
Aggregation	Weighted sum	Non-missing components only
Tie-breakers	Distance, brand/operator equality, raw name sim	Deterministic order

Table 3.2: Final RandomForestClassifier hyperparameters.

Hyperparameter	Value	Notes
n_estimators	650	Ensemble size (selected via grid search)
max_features	sqrt	Subspace sampling per split
max_depth	42	Depth cap controlling variance
min_samples_split	2	Allows deep branching
min_samples_leaf	2	Light regularisation vs. leaf=1
criterion	gini	Impurity metric
bootstrap	true	Bagging enabled
random_state	42	Reproducibility seed
n_jobs	-1	Full parallelism
Class weighting	(none)	No custom weights
Feature count	147	Stable, ordered vector
Probability threshold	0.50	Operational decision boundary (balanced)

- **Imports & refresh:** one-off ATP/OSM ingests and refreshing materialized feature views.
- **Inspect:** list tables, feature view versions, and recent run metadata.
- **Maintain:** clean transient staging tables and related housekeeping.
- **Dev utilities:** sample rows to CSV/GeoJSON and seed tiny test databases.
- **Execution & safety:** run via `docker compose run --rm db python cli.py <subcommand>` or locally (DuckDB); same config precedence as Section 4.5.4. Imports skip unchanged snapshots unless `--force`; migration guards prevent ad-hoc schema drift.

- **Out of scope:** diff generation, model training, and artifact export remain in their dedicated services to preserve auditing.

3.7 Validation & Evaluation

This section details the empirical methodology used to assess the effectiveness and efficiency of the proposed conflation workflow. We progressively (i) construct and use a curated golden dataset, (ii) enforce a high-level semantic categorization as a hard preselection (blocking) stage, (iii) fine tune the legacy rule-based matcher (PT1/PT2 lineage) via a grid-search style exploration, (iv) evolve an Random Forest Classifier-based matcher through two training regimes, (v) define the evaluation metrics and validation protocol, and (vi) compare the rule-based and machine learning (ML) approaches using identical golden references. Emphasis is placed on reproducibility, isolation of improvements, and avoidance of overfitting to narrow brand-specific patterns.

3.7.1 Golden Dataset

The golden dataset serves as the authoritative reference for external validation and model selection. It consists of two complementary components:

1. A stratified random sample of 200 POIs (“Random Sample”) drawn from Swiss ATP-sourced candidates spanning multiple categories, distance buckets, and urban/rural regions.
2. A domain-specific brand subset for *Aldi Süd Switzerland* (“Aldi Süd CH”). The dataset size is 246 records.

Sampling Protocol. For the 200-sample subset we first partition candidate pairs by high-level category (cf. § 3.7.2) and distance bucket (e.g., 0–25 m, 25–100 m, 100–500 m, 500–1’500 m) to avoid a proximity bias, then proportionally sample within each stratum. This guards against overrepresentation of dense urban groceries or petrol stations.

Annotation Workflow. Annotators labeled candidate pairs (match vs. non-match) using the OSM web interface and auxiliary map layers for context. Disagreements are adjudicated in a consensus pass with explicit rationale logging. Ambiguous brand co-location (e.g., shared building footprints) triggers an additional check of opening hours, address tokens, and name variants.

Usage. The golden dataset is *not* used in ML training folds; it is reserved strictly for: (i) early sanity checks of blocking misclassifications, (ii) threshold calibration for the rule-based matcher, and (iii) final side-by-side evaluation.

3.7.2 High-Level Categorization as Hard Blocking

Prior to feature computation, each incoming POI from OGD and candidate POIs from OSM is mapped to a high-level semantic category ("superclass"). Candidate generation is restricted to pairs sharing the same superclass. This constitutes a deterministic blocking strategy (hard filter) that reduces the combinatorial explosion of pairings, lowering false positives and computational cost [1, 10]. Superclasses cluster frequent OSM tagging namespaces (e.g., `amenity=restaurant`, `shop=supermarket`, `tourism=hotel`) into domain aggregates (e.g., gastro, retail, accommodation). The taxonomy is extensible to absorb emerging categories with minimal refactoring.

Observed coverage. After refinement with Swiss ATP data, 23'445 global POIs (7'460 Switzerland) remained categorized as *other* due to sparse or ambiguous tagging (e.g., airports, GBFS feeds).

3.7.3 Rule-Based Matcher Fine-Tuning

The rule-based matcher combines multiple similarity signals (e.g., name, address, geometry, brand/operator) via feature weights and a global similarity threshold. In the grid search reported here, only method toggles, feature weights, and the *similarity threshold* were varied; the maximum candidate distance was **fixed at 1,500 m** in all runs and was not tuned.

Objective. Rather than rating entire configurations, this analysis isolates the effect of *individual* comparators and weight patterns: which methods and weightings correlate with strong F1 scores ("good") and which with noticeable degradation ("bad")?

Threshold effect. Across 1,200 configurations, the global similarity threshold drives the usual precision–recall trade-off. A lower threshold (0.800) boosts recall but harms precision and increases error counts; thresholds in the 0.850–0.875 range yield the best average F1.

Table 3.3: Similarity threshold vs. average metrics (all 1,200 configurations)

Similarity thr.	n	F1	Precision	Recall	Avg. errors
0.800	240	0.9297	0.9007	0.9631	47.2
0.825	240	0.9380	0.9239	0.9543	40.8
0.850	240	**0.9416**	0.9441	0.9401	**37.7**
0.875	240	0.9409	**0.9558**	0.9270	**37.7**
0.900	240	0.9367	0.9612	0.9136	40.0

Comparators: patterns across runs. *Name* comparators show the largest spread: `jaro_winkler` tends to deliver higher recall but noticeably lower precision; `token_sort_ratio` and `levenshtein` are more balanced and yield the best F1 on average. For *address*, `jaccard` has a slight edge.

Table 3.4: Comparators (global averages across all configurations)

Field: name	n	F1	Precision	Recall
levenshtein	400	**0.9392**	**0.9503**	0.9292
token_sort_ratio	400	0.9389	0.9497	0.9293
jaro_winkler	400	0.9341	0.9114	**0.9604**

Field: address	n	F1	Precision	Recall
jaccard	600	**0.9375**	0.9378	0.9392
token_sort_ratio	600	0.9373	0.9365	**0.9400**

Weights: Top-20 vs. Bottom-20. The 20 best configurations (by F1) differ most from the 20 weakest in weights for *name*, *opening_hours*, and *website*. Higher *name* weight and down-weighting sparse fields correlate with better performance.

Table 3.5: Average feature weights (*Top-20* vs. *Bottom-20*)

Feature	Top-20 avg.	Bottom-20 avg.
geom	0.28	0.28
name	**0.22**	0.16
address	0.10	0.10
brand	0.10	0.10
operator	0.05	0.05
phone	0.05	0.05
tag_overlap	0.14	0.16
opening_hours	**0.03**	0.05
website	**0.03**	0.05

Concrete patterns (*good* vs. *bad*).

- **Good:** similarity threshold in 0.850–0.875; *name* comparator `token_sort_ratio` or `levenshtein`; *address* with `jaccard`; *name* weight $\geqslant 0.20$ (often 0.30 in top runs); *opening_hours* and *website* near zero.
- **Bad:** similarity threshold $= 0.800$ (bottom runs concentrate here; high recall but weak precision and more errors); *name* with `jaro_winkler` combined with a low threshold; over-weighting sparse fields (*opening_hours*, *website*); slightly inflated *tag_overlap* without measurable gain.

Takeaway. A precision-leaning operating point emerges around thresholds of 0.85–0.875, a *name* comparator that balances precision and recall (`token_sort_ratio` or `levenshtein`), and explicit down-weighting of sparse attributes. The fixed 1,500 m radius serves as a coarse candidate filter; most discriminative power comes from text comparators and their weighting.

3.7.4 ML-Based Matcher Evolution

We adopt a supervised Random Forest Classifier ([19, 20]) trained on engineered similarity and context features (147 dimensions, per logs). Two distinct training regimes were executed.

Phase 1 (Artificial Perturbations Only). Initial training relied solely on synthetic (artificially altered) pairs generated from OSM POIs (v2 run). When deployed on the Aldi Süd CH golden subset the model produced: Precision 0.9637, Recall 0.8230, F1 0.8878, False Positive Rate (FPR) 0.0363 (Real metrics). High precision but recall degradation indicates domain shift between synthetic perturbations and real-world divergences (e.g., partial brand names, evolving store metadata) leading to under-detection.

Phase 2 (Augmented with Rule-Based Harvest). The improved pipeline (v7 run) supplements synthetic data with additional weakly labeled pairs harvested via high-precision rule-based matches, then filtered, before stratified train/test splitting: Training set 68'573 instances, Test set 17'144 (feature dimension validated at 147). Grid search over 36 hyperparameter combinations yields best accuracy 0.9643 with deep trees (max depth 42, 650 estimators, minimal leaf/split values). Applied to the Aldi Süd CH golden subset: Precision 0.9957, Recall 0.9914, F1 0.9935, FPR 0.00433—eliminating the earlier recall gap while further reducing false positives.

Table 3.6: Aldi Süd CH: ML Evolution (Real Metrics)

Phase	TP	FP	FN	Precision	Recall	F1
v2 (Artificial Only)	186	7	40	0.9637	0.8230	0.8878
v7 (Augmented)	230	1	2	0.9957	0.9914	0.9935

Observed Gains. Relative F1 improvement: $(0.9935 - 0.8878)/0.8878 \approx 11.9\,\%$. False negatives reduced from 40 to 2 (95 % reduction), indicating successful coverage expansion.

3.7.5 Evaluation Metrics

We report the following metrics on held-out golden data (and internal validation splits for ML):

- Precision, Recall, F1-Score: Primary effectiveness indicators balancing omission (FN) and commission (FP) errors [6].
- False Positive Rate (FPR) = FP / (FP + TN): Critical to avoid introducing incorrect edits into OSM.
- Throughput (records/second): Operational scalability (measured end-to-end excluding network I/O variability).

The thresholds for deployment are chosen to maximize F1 subject to an upper bound on FPR (operational policy).

3.7.6 Rule-Based vs. ML-Based Matcher

We contrast the tuned rule-based matcher (best configuration cluster) with the evolved ML model (run v8) further trained with additional training data using identical golden subsets.

Training Data Differences.

- Rule-Based: Deterministic similarity functions + globally tuned weights/thresholds; no statistical fitting beyond grid search enumeration; relies directly on golden data only for evaluation and threshold sanity checks.
- ML-Based Phase 1: Trained exclusively on synthetic perturbations; mismatch with real variation caused recall deficit.
- ML-Based Phase 2: Incorporates additional high-precision pseudo-labeled pairs from rule-based matches (semi-supervised flavor) + synthetic pairs + internal stratified split; golden data held out strictly for external validation.

Comparative Metrics (Aldi Süd CH). Focusing on the tuned rule-based matcher versus the ML variants, the table shows a clear precision-first trend for ML while keeping recall essentially unchanged. On Aldi Süd CH, the rule-based approach reaches Precision 0.9914 (two false positives), Recall 0.9957 (one false negative), and F1-Score 0.9935. ML v7 already reduces spurious matches (Precision 0.9957 with one false positive) with a minor recall change (0.9914) and the same F1-Score 0.9935. ML v8 removes spurious matches entirely (Precision 1.0000), keeps Recall at 0.9957, and yields the best F1-Score 0.9978.

Complementary Sample (Random 200). The same pattern appears on the class-agnostic sample: compared to the rule-based baseline (Precision 0.8495, Recall 0.8681, F1-Score 0.8587), ML v8 lifts precision to 0.9398 with a modest recall decrease to 0.8298, resulting in a higher F1-Score 0.8814.

Table 3.7: Golden-set comparison (Aldi Süd CH and Random 200): confusion matrices and derived metrics.

Dataset	Approach	TP	FP	FN	TN	Precision	Recall	F1	FPR
Aldi Süd CH (246)	ML (PT1 baseline)	208	37	0	1	0.8490	1.0000	0.9183	0.9737
Aldi Süd CH (246)	Rule-Based (tuned)	230	2	1	13	0.9914	0.9957	0.9935	0.0086
Aldi Süd CH (246)	ML (v7)	230	1	2	13	0.9957	0.9914	0.9935	0.0043
Aldi Süd CH (246)	ML (v8)	232	0	1	13	1.0000	0.9957	0.9978	0.0000
Random 200 (200)	Rule-Based (tuned)	79	14	12	95	0.8495	0.8681	0.8587	0.1505
Random 200 (200)	ML (v7)	76	6	17	101	0.9268	0.8172	0.8686	0.0732
Random 200 (200)	ML (v8)	78	5	16	101	0.9398	0.8298	0.8814	0.0602

Scalability. ML inference cost remained acceptable; however, a direct runtime comparison table across increasing dataset sizes is pending.

Table 3.8: Runtime by dataset and approach (wall-clock seconds from logs).

Dataset Size (POIs)	Approach	Runtime (s)
246 (Aldi Süd CH)	Rule-Based (tuned)	3.45
246 (Aldi Süd CH)	ML (v7)	2044.37
246 (Aldi Süd CH)	ML (v8)	1899.75
200 (Random 200)	Rule-Based (tuned)	4.63
200 (Random 200)	ML (v7)	1584.33
200 (Random 200)	ML (v8)	1502.24

3.8 Discussion & Limitations

3.8.1 Interpretation of Results

The progression from legacy rule-based heuristics to tuned weights and then to a multi-feature Random Forest Classifier yields a monotonic precision improvement on the brand-focused subset while retaining near-ceiling recall after initial tuning. Precision gains correspond to pruning residual lexical and geographic false positives; recall stabilization suggests a diminishing pool of structurally hard negatives (e.g., alias drift, incomplete addresses). On the heterogeneous Random 200 sample, ML variants consciously trade a modest recall decrease for a substantive False Positive Rate contraction—an acceptable trade-off when minimizing harmful insertions into OSM has priority. Semantic superclass blocking meaningfully lowers candidate volume, concentrating matcher effort on plausible pairs and lowering FP risk through early elimination.

Comparative Metric Trajectories. Precision increases stepwise (baseline → tuned rule-based → ML v7/v8) while F1 consolidates in a high-performance band. The False Positive Rate collapse reflects both better discrimination and improved true negative counting after instrumentation adjustments. Residual errors cluster in categories with sparse or noisy metadata (e.g., mixed-use buildings, recently rebranded outlets).

Error Surface Evolution. The staged approach—(i) threshold/weight calibration, (ii) synthetic+weakly labeled augmentation, (iii) iterative retraining—shrinks the joint FP+FN error surface. Gains increasingly arise from targeted feature refinements (e.g., alias normalization, geometry snapping) rather than global hyperparameter shifts, indicating diminishing returns of generic grid search.

3.8.2 Typical Failure Modes

- **False positives (FP).** Co-located chain outlets (e.g., discount grocery adjacent to a logistic pickup point) sharing high name similarity and overlapping geometry; generic brand tokens (“Express”, “Center”) spuriously aligned.

- **False negatives (FN).** Multilingual or legacy brand variants (“Aldi Suisse” vs. “Aldi Süd”); address drift where house numbers changed post-renovation; POIs tagged under alternative but semantically similar categories resulting in blocking exclusion.
- **Sparse metadata.** Source entries lacking street names or brand tags reduce discriminative power; in such cases geometry+name text alone insufficiently separates near duplicates.
- **Geometric anomalies.** Large building polygons with centroid offsets produce inflated distance-derived penalties; indoor mall units collapsed to a shared entrance node.
- **Temporal staleness.** Recently closed or newly opened stores produce asymmetric freshness between ATP snapshots and OSM edits, yielding transient mismatches.

Illustrative Examples. *FP:* Two adjacent branded petrol stations where operator tags differ but brand tokens match; model overweights name similarity. *FN:* A store renamed locally (shortened alias) while upstream data retains the extended name; token reordering plus missing house number reduce combined similarity below threshold.

3.8.3 Generalisability

The pipeline generalizes to additional countries and POI verticals given: (i) extension of the superclass taxonomy; (ii) retraining with geographically and linguistically representative samples; and (iii) recalibration of decision thresholds respecting new False Positive cost structures. Domain shift (different addressing schemes, cultural naming patterns) can degrade recall when synthetic perturbations fail to capture locale-specific variance; incorporating a modest seed of real annotated pairs from the target domain mitigates this risk.

3.8.4 Runtime and Scalability

Current ML inference exhibits substantially higher wall-clock time than the tuned rule-based approach due primarily to feature vector construction and evaluation of a large ensemble of deep trees. Early optimization opportunities include: (i) pruning uninformative or redundant features (guided by permutation importance / SHAP); (ii) batching similarity computations with vectorized primitives; (iii) replacing deep forests with calibrated gradient boosting or distilled shallow ensembles; (iv) caching deterministic text similarity scores for repeated tokens. Given that semantic blocking already thins the candidate pool, marginal runtime gains have leverage in large-scale periodic reruns (tens of thousands of POIs).

3.8.5 Mitigation Strategies for Failure Modes

- **Alias expansion.** Maintain a curated mapping of brand and chain aliases (locale variants) injected pre-matching to lift recall without lowering precision.

- **Adaptive thresholds.** Calibrate per-superclass thresholds to reflect heterogeneous FP risk (e.g., higher threshold for chains with dense urban clustering).
- **Geometry normalization.** Snap large polygon centroids to entrances or compute minimum distance between polygon edges to reduce artificial distance inflation.
- **Active error harvesting.** Periodically sample near-threshold non-matches and confirmed mismatches for annotation to focus augmentation on high-uncertainty regions.

3.8.6 Limitations

- **Golden set scope.** The golden dataset size (brand-focused + 200 heterogeneous pairs) bounds statistical confidence; rare categories remain under-sampled.
- **Synthetic bias.** Artificial perturbations may under-represent real-world noise (e.g., abbreviations, transliteration issues), potentially inflating internal validation scores.
- **Runtime overhead.** Current ML latency may constrain interactive auditing; absence of streaming or incremental inference increases end-to-end cycle time.
- **Explainability.** While feature importances exist, deep ensemble interactions hinder fine-grained rationale extraction for individual decisions compared to explicit rule weights.
- **Data drift monitoring.** No automated drift detection (e.g., population stability index) is yet integrated; degradation risk between scheduled retraining cycles persists.

3.8.7 Ethical & Licensing Considerations

Respect ODbL provenance and source attribution; retain a human-in-the-loop via auditing tools (e.g., OSM Conflator and cf_audit) for contentious merges; avoid automatic application of low-confidence matches. Logging of decision features must exclude personal data beyond what is already public in OSM. For external brand datasets, ensure redistribution terms permit derivative conflated outputs.

3.8.8 Future Work Direction (Synthesis)

Focus shifts from broad metric elevation to targeted robustness: multilingual normalization, per-category adaptive thresholds, probabilistic calibration (e.g., Platt scaling) for triage, and light-weight model distillation for runtime reduction. Integrating drift dashboards and automated retraining triggers would complete an MLOps feedback loop.

3.9 Section Summary

- **Key findings (effectiveness).** Superclass (semantic) blocking markedly reduces false positives and compute by constraining candidate pairs; tuned rule-based matching achieves high F1, and an augmented Random Forest Classifier further elevates precision while sustaining near-ceiling recall on brand-focused data and reducing False Positive Rate on heterogeneous samples.
- **Key findings (efficiency).** ML inference latency remains higher (feature construction + large ensemble) but is acceptable for batch conflation; optimization potential lies in feature pruning, vectorization, and model distillation.
- **Data curation.** A stratified golden dataset (brand-specific + heterogeneous sample) enabled unbiased external validation and threshold calibration while avoiding training leakage.
- **Rule-based tuning insights.** Optimal similarity thresholds cluster in the 0.85–0.875 band; overweighting sparse fields (website, opening_hours) degrades performance; balanced name comparators (Levenshtein, token_sort_ratio) dominate top configurations.
- **ML evolution insights.** Augmenting synthetic perturbations with high-precision pseudo-labeled matches closes recall gaps and suppresses residual false positives; marginal future gains likely require targeted feature engineering over broader hyperparameter search.
- **Error characteristics.** Remaining errors concentrate in sparse-metadata, alias-variant, or co-located multi-brand scenarios, suggesting focused alias expansion and geometry normalization will yield outsized returns.
- **Operational lessons for DiffedPlaces.** Maintain idempotent ingestion in DuckDB; cache deterministic similarity scores; surface per-category metrics to support adaptive thresholds; expose auditable diffs via osm_conflate / cf_audit to keep a human review loop.
- **Quality governance.** Enforce external validation strictly separated from training folds; implement drift monitoring (e.g., distribution shift alerts) before expanding geographic scope.
- **Limitations.** Golden set coverage of rare categories is thin; runtime overhead of the ensemble limits interactive usage; absence of probabilistic calibration constrains nuanced triage.
- **Next steps.** (i) Integrate alias dictionary + multilingual normalization; (ii) add per-superclass adaptive thresholds and probabilistic calibration; (iii) implement feature importance-driven pruning and model distillation; (iv) deploy drift dashboards and automated retraining triggers; (v) extend annotated golden data for underrepresented categories.

Chapter 4

Software Project Documentation

4.1 Vision

The overarching vision of the software part of this thesis is to provide a reproducible, modular and extensible geospatial data conflation service—the DiffedPlaces service—capable of continuously identifying, classifying, and reconciling differences between heterogeneous POIs datasets (e.g., OSM and external sources such as ATP) while minimizing manual curation effort. Building on conceptual and experimental insights from [12, 13], the software aims to close the gap between experimental matching pipelines and an operational foundation that can: (i) generalize matching logic across categories and jurisdictions, (ii) surface high-quality candidate updates for human verification, and (iii) feed improvements back into open data ecosystems.

Key aspirational qualities are:

- **Generality**: A unified abstraction that supports multiple external source spiders (initially ATP) without code duplication.
- **Traceability**: Every produced *diff* item is reproducible from versioned inputs (source snapshots, configuration, model versions).
- **Incremental operation**: Ability to run periodic refresh cycles ingesting only deltas where feasible.
- **Quality awareness**: Embedded evaluation instrumentation (precision / recall / F1-Score) to monitor drift of rule-based and machine learning components.
- **Human-in-the-loop readiness**: Outputs structured for subsequent auditing (e.g., via future OSM Auditor style tooling).
- **Sustainability**: Lightweight dependencies (e.g., DuckDB for analytical queries) combined with scalable components to enable both local research usage and potential community deployment.

The vision explicitly separates concerns between data acquisition, normalization, feature engineering, matching, diff generation, and audit/export to encourage evolvability of individual stages without systemic rewrites.

An intentionally lightweight, iterative (agile-inspired) delivery approach is embraced: incremental vertical slices (e.g., a single retailer end-to-end) are completed and inspected before broad generalization, reducing rework and enabling earlier validation of assumptions. Formal heavyweight artifacts are replaced by evolving configuration, concise architectural notes, and empirical evaluation logs.

4.2 Requirements

This section distills functional and non-functional requirements derived from the thesis objectives, prior projects [12, 13], and the operational constraints of the DiffedPlaces service.
Requirements were not frozen up-front; instead they evolved iteratively based on feasibility insights from early vertical slices and feedback gathered during biweekly reviews. Adjustments (e.g., refining feature engineering scope, deferring real-time ingestion) were captured directly in configuration and golden dataset expansion rather than in a separate change request process.

4.2.1 Functional Requirements

1. **Ingest External Source Data**: Support fetching and parsing ATP (initial set) and be extensible to additional open datasets.
2. **Ingest OSM Baseline**: Acquire relevant OSM extracts (bbox, country, or thematic) and normalize them into a common internal schema.
3. **Normalization and Cleaning**: Standardize names, addresses, categories, and coordinates.
4. **Feature Engineering**: Compute textual similarity features (e.g., token ratios, Jaro–Winkler), spatial features (distance, bearing), and categorical compatibility indicators for each candidate pair.
5. **Blocking / Candidate Generation**: Efficiently generate candidate POI pairs using spatial indexing (distance/radius queries) and heuristic category constraints to bound complexity.
6. **Rule-based Matching**: Provide deterministic baseline rules combining distance thresholds and string similarity to classify matches / non-matches / possible matches.
7. **Machine Learning Matching**: Train and apply one or more supervised models (e.g., Random Forest, Gradient Boosting, LightGBM) using engineered features to improve recall/precision balance.
8. **Model Evaluation**: Compute precision, recall, and F1-Score over curated golden datasets; retain historical evaluation snapshots for regression detection.

9. **Diff Generation**: Produce structured diff artefacts enumerating create / update / delete / ambiguous actions relative to OSM baseline.
10. **Export for Auditing**: Output machine- and human-readable artefacts (e.g., CSV / GeoJSON) suitable for downstream auditing and potential ingestion into an OSM Auditor workflow.
11. **Configuration Management**: Centralize thresholds, weights, and feature toggles in JSON configuration enabling reproducible experiments.
12. **Command Line Interface**: Provide scripted entry points for ingest, feature generation, training, inference, and diff publication to enable automation.
13. **Logging and Traceability**: Log processing steps with input, configuration version, and model identifier for each run.
14. **Golden Data Tooling**: Support generation and maintenance of golden datasets (manual curation assisted by heuristics) for ongoing evaluation.
15. **Error Handling**: Gracefully handle network timeouts, malformed records, and missing attributes while flagging impacted diff items for review.

4.2.2 Non-Functional Requirements

- **Performance**: Process typical national-scale category subsets (e.g., a retailer chain) within a practical runtime on commodity hardware .
- **Scalability**: Architectural separation allows horizontal scaling of candidate generation and feature computation if data volume grows.
- **Modularity**: Clear separation between ingestion, matching logic, and diff rendering to permit algorithm substitution.
- **Extensibility**: Adding a new source should require only implementing source-specific extraction and minimal mapping configuration.
- **Reproducibility**: Re-running with identical inputs and configuration must yield identical diff outputs.
- **Observability**: Metrics (counts of candidate pairs, match class distribution, evaluation scores) are exposed or logged for monitoring.
- **Portability**: Minimal external dependencies; ability to run via containerization for consistent environments.
- **Licensing Compliance**: Preserve ODbL attribution considerations when generating outputs that reflect OSM derived data.

- **Maintainability**: Code follows consistent naming, documentation headers, and lightweight test coverage for core functions.

4.2.3 Assumptions and Constraints

- Availability of periodically updated ATP snapshots and accessible OSM extracts (e.g., via planet extracts subset).
- Golden datasets are of limited size; thus model generalization must cope with class imbalance.
- System operates in a batch mode; real-time updates are out of scope for this iteration.
- Manual auditing capacity is limited; precision is prioritized over exhaustive recall for high-impact categories.

4.2.4 Out of Scope

- Direct automated editing of OSM; outputs remain advisory.
- Full-fledged web-based auditing platform (future integration point with OSM Auditor).
- Global scale full planet conflation in a single batch.
- Integration of a scheduler for regular automated runs.

4.3 Analysis

The analysis focuses on the characteristics of the source datasets, the nature of POI identity resolution, and constraints influencing architectural and algorithmic choices.

4.3.1 Data Sources Characteristics

OpenStreetMap. OSM provides volunteered geographic information (VGI) with heterogeneous tagging practices and varying positional and categorical accuracy. Name fields may include branding variants, language-specific tokens, or disambiguators (e.g., city district) that require normalization. Geometry is predominantly node-based for many commercial POIs, but polygons and multipolygons occur for large facilities.

All The Places. ATP aggregates structured retailer/location data via spiders; it tends to offer more uniform attribute presence for specific brands (e.g., opening hours, operator names) but may lack harmonized category tags relative to OSM. Geometries are usually single latitude/longitude points. Address formatting can differ (abbreviations, punctuation) and may require canonicalization.

4.3.2 Entity Resolution Challenges

- **Ambiguous Naming**: Chains whose brand equals a common word increase false positives.
- **Near-Duplicate Clusters**: Shopping centers may contain multiple similar branded sub-entities (e.g., pharmacy counters) close to a main store.
- **Temporal Drift**: Store relocations cause stale OSM entries offset from updated external coordinates.
- **Category Divergence**: Tagging granularity mismatch (e.g., `shop=supermarket` vs. external classification taxonomy) complicates category-based blocking.
- **Sparse Addresses**: Missing housenumbers reduce discriminative power of address similarity metrics.

4.3.3 Similarity Feature Rationale

Textual similarity leverages robust string metrics (token set ratio, partial ratio, Jaro–Winkler) to accommodate transpositions and minor spelling discrepancies. Geospatial distance provides a strong negative signal beyond an empirically derived radius of 1'500 meters, which was determined as necessary to capture all matches during the creation of the golden dataset. Category compatibility acts as a prior: improbable mappings are penalized. Additional derived signals (e.g., shared address tokens count, normalized brand canonical form equality) further refine candidate scoring.

4.3.4 Blocking Strategy Considerations

Exhaustive pairwise comparison is infeasible at scale (quadratic growth). Spatial proximity filtering (distance/radius queries and lightweight spatial indices) constrains candidate generation while preserving recall. See Section 3.4

4.3.5 Golden Data and Evaluation

Accurate supervised learning requires curated positive and negative examples. An iterative golden dataset refinement loop was adopted: initial heuristic matches were reviewed manually, false positives and negatives were annotated, and the resulting labeled pairs were reused for model training.

The initial sampling for this dataset was based on the `aldi_sud_ch` spider output, which provided a consistent set of POIs from the ATP dataset. From this pool, exactly 200 POIs were selected using Python's `random_sample` function, optionally seeded for reproducibility. This ensured an unbiased, geographically diverse subset while allowing the sampling process to be repeated identically for debugging or incremental dataset expansion. The codebase implements this logic directly, making the process transparent and modifiable for future sampling strategies.

To mitigate class imbalance, strategies such as undersampling of easy negatives and feature-threshold guided sampling were considered and partially applied during dataset generation. This loop aligns with an agile mindset: each iteration treats the enriched golden dataset plus updated evaluation metrics as an inspect-and-adapt checkpoint, informing subsequent prioritization—such as introducing additional similarity features, adjusting sampling parameters, or broadening retailer coverage.

4.3.6 Risk and Complexity Drivers

- **Data Volume Variability**: Different categories produce widely varying candidate densities.
- **Model Generalization**: Overfitting to a single retailer brand jeopardizes performance on unseen chains.
- **Operational Reproducibility**: Without strict configuration/version control, diff reproducibility degrades.
- **Human Review Load**: Excess ambiguous candidates inflate auditing effort.

These drivers motivate a design emphasizing modular feature computation, traceable configurations, and tunable thresholds to balance precision and recall.

4.4 Design

The design follows a layered, pipeline-oriented architecture that separates acquisition, normalization, matching, and diff materialization. This enables independent evolution of algorithms and data connectors while preserving reproducibility.

4.4.1 High-Level Architecture

At a high level the system consists of:

Acquisition Layer Source-specific spiders / fetchers for ATP (extendable) and retrieval of OSM extracts. Outputs raw snapshot files.

Normalization Layer Transforms raw records into a canonical internal schema: identifiers, geometry, category mapping, tags/properties.

Feature Engineering Layer Generates similarity features for candidate pairs (textual, spatial, categorical, address-based, brand equality flags).

Matching Layer Applies rule-based filters then machine learning classifiers to assign labels (match / non-match / ambiguous) with associated confidence.

Diff Assembly Layer Aggregates labeled pairs to construct proposed create/update/delete actions relative to OSM. Supports export to CSV / GeoJSON and optional ChangeXML for bulk editing.

Evaluation and Reporting Computes metrics on golden datasets and logs run metadata.

4.4.2 Component Responsibilities

- **Configuration & Orchestration**
  - Centralizes run-time settings (distance thresholds, enabled features, model identifiers) from environment variables and optional JSON configuration.
  - Lives alongside the Docker Compose stack and provides sane defaults for local development and repeatable experiments. No internal scheduler is bundled; periodic execution is expected to be triggered externally (e.g., cron, CI), see *Out of Scope*.
- **DB Service (QuackOSM + DuckDB)**
  - Imports and harmonizes OSM extracts and ATP spider outputs into a unified POI schema; enforces schema-version tags on every table and view.
  - Provides high-throughput analytical queries (spatial filtering, candidate generation, feature materialization) via DuckDB, optimized for in-memory columnar processing.
  - Maintains spatial indices to bound candidate search (distance/radius queries and bounding-box partitioning) and exposes reusable SQL views consumed by the Diff service.
- **Diff Service**
  - Orchestrates end-to-end conflation for a given spider: data pull → candidate generation → similarity scoring → match decision → diff creation.
  - Integrates `osm_conflate` for producing human-reviewable `.osm` and `.json` diffs; bundles trace metadata (run id, config hash, model version) into every artefact.
  - Hosts the HTTP endpoints used in practice:
    * `/run_diff/<spider_name>` – generate diffs for a specific spider.
    * `/recheck_diff/<spider_name>` – optional re-audit of edited diffs.
- **Download Service (Web UI/API)**
  - Lightweight Flask application that lists and serves produced diffs for inspection and download.
  - Acts as a public access point for auditors and downstream tooling; surfaces per-run provenance (run fingerprint, input dataset summaries, counts of TP/FP/FN when available).

- **Library Module (Shared Utilities)**
  - Encapsulates string normalization (case/diacritics folding, punctuation handling), tokenization, and standard string similarity functions (e.g., token-set/partial ratios, Jaro–Winkler).
  - Provides geodesic distance, category mapping helpers, brand canonicalization, and feature engineering primitives used by both rule-based and ML matching.
  - Centralizes error handling, logging, and metrics emission to keep the services thin and consistent.
- **ML Module**
  - Defines the feature schema, trains the classifier, and persists the full inference bundle (estimator + feature list + preprocessing steps + version).
  - Exposes batch inference APIs to the Diff service; supports score calibration and configurable decision thresholds to trade precision/recall per source.
  - Stores model cards and evaluation reports alongside artefacts to document training data, metrics, and known limitations.
- **Import APIs**
  - Simple HTTP entry points to populate the DB Service:
    * `/import_osm` – ingest OSM extracts into DuckDB.
    * `/import_atp` – ingest ATP spider dumps into DuckDB.
  - Validate payloads against the unified POI model and stamp schema/version metadata.
- **Export Module**
  - Renders diff outputs (`.osm`, `.json`) and auxiliary audit artefacts (per-run logs, summary tables) with complete provenance (run id, config hash, model id, input dataset digests).
  - Publishes artefacts to the Download Service and optionally to external object storage for long-term retention.

4.4.3 Iterative Architectural Validation

Rather than finalizing all components up-front, the system was validated incrementally: a minimal viable path (single-brand ingestion $\mathrm{aldi\_sud\_ch} \rightarrow$ diff export) established the core abstractions; additional spiders and features were then integrated to test extensibility.

Refactorings from early runs. (a) The export was consolidated to a single *result.json* plus an *.osm* diff per run, written under `diffed_places/<spider>/`, and supplemented by inspection artefacts for

matched and *unmatched* features; exports now preserve raw OSM tags (no “beautification”). (b) The rule-based matcher evolved (v3→v4) with a tightened category map (e.g., `car`, `car_repair`, `car_wash` ⇒ *retail*) and a fallback from `name` to `brand/operator` for divergent labels; return types were made explicit. (c) The ML-Based Matcher path was gated initially while precision was improved; integration proceeds as additional labeled data become available.

4.4.4 Batch Data Flow and Artefacts

The typical batch run proceeds through clearly separated data layers and aggregation steps:

1. **Acquire source snapshots (Raw Layer):** fetch OSM extracts, ATP spider dumps, and optional OGD payloads.
2. **Normalize to Canonical Layer:** load into the unified POI schema (DuckDB/QuackOSM), stamp schema & run metadata.
3. **Candidate Generation (Blocking):** spatial proximity filtering (radius / bounding-box strategies).
4. **Feature Engineering:** compute textual, spatial, and categorical features for each candidate pair.
5. **Matching:** classify pairs via Rule-Based or ML-Based Matcher; pass uncertain pairs forward.
6. **Aggregation:** consolidate labels into reconciliation actions (create/update/skip) for diff construction.
7. **Metrics:** if golden labels exist, compute precision/recall/F1 and runtime statistics.
8. **Export (Artefacts Layer):** write human-reviewable *JSON diff* and *.osm* diff for auditing; publish inspection artefacts (matched/unmatched GeoJSON) and metrics.

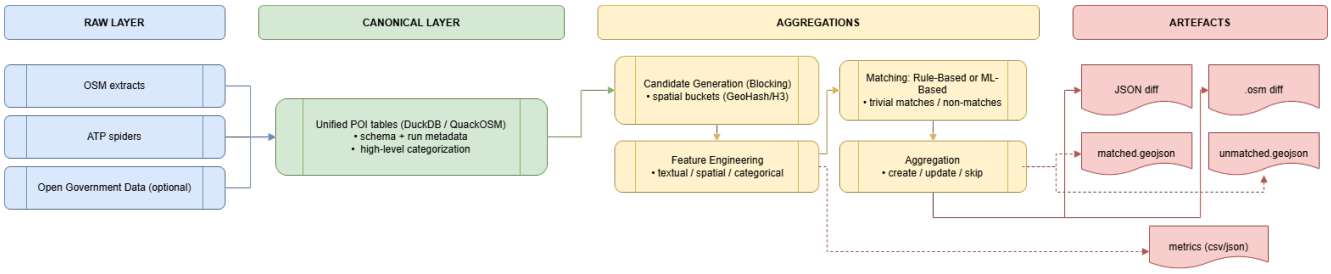

Figure 4.1: End-to-end data flow showing data layers (Raw, Canonical, Artefacts), aggregation steps (blocking, features, classification, aggregation), and resulting exports (JSON diff, .osm diff, matched/unmatched GeoJSON, metrics).

4.4.5 Configuration Strategy

Configurations are version-controlled alongside code to guarantee reproducibility. Each run records: configuration hash, data snapshot identifiers, and model version. Parameter groups include: distance thresholds, textual similarity cutoffs, feature inclusion flags, model hyperparameters.

4.4.6 Extensibility Mechanisms

The system accepts new data sources, additional ATPs spiders, and new countries with minimal friction by enforcing clear contracts at each layer and separating the most common onboarding use cases.

Shared contract (applies to all new inputs).

- **Unified POI model**: ingestors must yield WGS84 lon/lat and the canonical fields (`id`, `name`, `brand/operator`, `addr_*`, `phone`, `website`, `category_raw`, `geom`, `source`, `country_code`, `updated_at`). Adapters write to DuckDB via the DB service API.
- **Category mapping**: extend the dictionary-based mapper in the Library module to translate `category_raw` into high-level classes (*retail*, *gastro*, *health*, *transport*, ...), with unit tests.
- **Normalization hooks**: reuse shared utilities for case/diacritics folding, tokenization, address parsing, brand canonicalization, and phone/URL cleanup.

A) Adding a new ATP spider (dataset already in ATP, e.g., CH).

- **Steps**: (1) enable the spider in the run configuration; (2) call `/import_atp` to load the dump into the unified schema; (3) extend category mapping if new tags appear; (4) trigger `/run_diff/<spider_name>` in the Diff service; (5) review *result.json* (JSON diff), the *.osm* diff, matched/unmatched GeoJSON and metrics.
- **Code impact**: typically configuration-only; mapper tweaks if unfamiliar tags/brands surface.

B) Dataset *not* yet in ATP (become a new ATP spider first).

- **Recommended path**: contribute an upstream spider to ATP (schema, licensing, scraping). This keeps a single ingestion strategy and benefits from ATP's update cadence and QA.
- **After upstreaming**: proceed as in (A): configure, `/import_atp`, adjust mappings if needed, then `/run_diff/<spider_name>`.

C) Adding a new OGD source (parallel pipeline with broader scope like ATP).

- **Importer (ATP-like)**: implement a lightweight importer that emits the unified POI model and expose `/import_<ogd>` in the DB service.

- **Reuse the core**: blocking, feature engineering, rule-based and ML matching operate unchanged on the unified schema.
- **Running diffs**: add a thin runner in the Diff service (e.g., `/run_diff/ogd/<dataset>`) or select the provider via configuration; exports mirror the ATP flow (*result.json*, *.osm* diff, matched/unmatched GeoJSON, metrics).
- **Why simple**: only the importer is new; downstream logic and auditor-facing artefacts are shared.

D) Extending the feature set (rule-based and ML).

- **Feature registry**: add pure derived-feature functions with a short registration (name, dtype, dependencies), e.g., *token-set ratio*, *Jaro–Winkler*, *brand_equal*, *addr_overlap_count*, *geodesic_distance_m*.
- **Pipelines**: materialize features in DuckDB views for reproducibility; the rule-based matcher consumes a subset, the ML matcher the full vector.
- **Versioning**: bump the feature schema version and persist the model bundle (estimator + preprocessing + feature list + version). Old models remain runnable against their pinned schema.

E) Enabling a new country.

- **Locale**: add language-specific token rules (stopwords, transliteration), brand aliases, and address normalizers (street abbreviations, postcode formats).
- **Spatial partitioning**: register the country boundary and choose an appropriate blocking resolution to keep candidate sets tractable.
- **Threshold profiles**: define per-country defaults for distance/name thresholds in the rule-based prefilter; keep them runtime-configurable.
- **Data import**: wire OSM extracts for the country into QuackOSM; ensure adapters set `country_code` for correct filtering and metrics.
- **Validation**: seed a golden sample (e.g., 200–1'000 pairs) to recalibrate precision/recall and, if enabled, refresh the ML model.

F) Operational wiring.

- **Configuration & traceability**: all knobs (blocking, thresholds, enabled features/model) live in config/env; every artefact is stamped with the run fingerprint (config hash + Git commit).
- **Testing**: each new adapter/mapping ships with fixtures and a smoke run (`/run_diff/<name>`) to verify end-to-end exports.
- **Scheduling**: periodic execution is external (cron/CI); no integrated scheduler is bundled.

4.4.7 Quality Controls

Automated unit tests cover normalization edge cases (address parsing, token cleaning), feature computations, and classifier prediction invariants (e.g., monotonicity when distance increases).

4.5 Implementation and Testing

Implementation consolidates the architectural concepts into a Python-based codebase (the DiffedPlaces service) emphasizing modular pipeline steps and containerized deployment.

4.5.1 Technology Stack

- **Language**: Python for rapid experimentation and rich geospatial/string processing ecosystem.
- **Data Handling**: DuckDB for analytical queries, standard GeoJSON for interchange.
- **String Similarity**: Libraries such as TheFuzz / JaroWinkler provide token-based and edit-distance metrics
- **Machine Learning**: scikit-learn (e.g., Random Forest)
- **CLI / Orchestration**: Python entry points with argument parsing to trigger pipeline stages.
- **Containerization**: Docker images encapsulate runtime dependencies for reproducibility.

4.5.2 Module Structure

Core modules align with the end-to-end design—data access, normalization, blocking/candidate generation, feature engineering, matching, evaluation, and diff export. Wherever practical, functions follow pure-input/pure-output semantics (no hidden I/O, deterministic outputs given inputs) to simplify unit testing and reproducibility. The implementation is deployed via a Docker Compose stack with three cooperating services.

Service-level architecture (Docker Compose).

- **DB Service (QuackOSM + DuckDB)** — Port `5001`. Ingests OSM extracts and ATP spider dumps into a unified POI schema and exposes import endpoints: `/import_osm`, `/import_atp`, and optional `/import_<ogd>`. Provides materialized SQL views for blocking and feature computation. Stateful: mounts a writable volume for database files and a read-only volume for source data.
  - *db/*: `db_app.py` (HTTP API), `data_access.py` (DDL/DML helpers), schema migrations, and SQL view definitions.

  - Responsibilities: schema/version stamping, run metadata persistence, spatial proximity filtering (distance/radius heuristics without fixed hash grids), candidate queries, and feature table materialization.

- **Diff Service** — Port `5002`. Orchestrates the conflation run for a given spider or provider: candidate pull → similarity scoring → match decision → diff creation. Stateless and horizontally scalable; safe to run multiple replicas. Endpoints include `/run_diff/<spider_name>` (and optionally `/run_diff/ogd/<dataset>`).
  - *diff/*: `diff_app.py` (HTTP API + pipeline), `match_rule.py` (rule-based matcher), `match_ml.py` (ML matcher), exporters.
  - Responsibilities: pipeline coordination, Rule-Based prefilter, ML-Based Matcher inference, aggregation to actions (create/update/skip), and export of `result.json` and `.osm` diff plus inspection artefacts (matched/unmatched GeoJSON, metrics).
- **Download Service (Web UI/API)** — Port `5003`. Lightweight Flask app listing produced artefacts under `diffed_places/` for audit and download; embeds per-run provenance (run fingerprint, input summaries, counts).
  - *download/*: `download_app.py` (UI/API), static listing, simple search.
  - Responsibilities: browse/download diffs, expose provenance and basic metrics to auditors.

Compose wiring. A shared user-defined network connects the three services. Volumes: (i) `data_dir` (read-only sources), (ii) `db_store` (DuckDB files), (iii) `diffed_places` (exports). Config is injected via env vars (e.g., `CONFIG_PATH`, thresholds, enabled features/model). No scheduler is bundled; periodic runs are triggered externally.

Code-level modules (inside services).

- **Configuration & run fingerprint** — reads JSON + env, resolves defaults.
- **Data Access** — typed accessors for DuckDB (reads/writes, parameterized SQL, pagination), schema migration helpers, and I/O of Parquet for caches or fixtures.
- **Normalization** — case/diacritics folding, punctuation and whitespace policy, brand/operator canonicalization, URL/phone cleanup, address tokenization, and country-aware normalizers.
- **Blocking / Candidate Generation** — spatial partitioning; supports per-source overrides of resolutions and block keys.
- **Feature Engineering** — pure functions that emit derived columns with a small registry (name, dtype, dependencies): token-set/partial ratios, Jaro–Winkler, `addr_overlap_count`, `brand_equal`, geodesic

distance (m), category compatibility, etc. Features are materialized as DuckDB views for reproducibility.

- **Matching**
  - *Rule-Based prefilter*: fast acceptance/rejection of trivial positives/negatives using calibrated name/distance/category thresholds; tunable per country/source.
  - *ML-Based Matcher*: batch inference using a persisted bundle (estimator + preprocessing + feature list + version) with score calibration and configurable cutoffs.
- **Aggregation & Diff Export** — converts labels to reconciliation actions and renders human-reviewable `result.json`, `.osm` diff, matched/unmatched GeoJSON, and metrics; every file embeds the run fingerprint.
- **Evaluation & Metrics** — if golden labels exist, computes precision/recall/F1 and false-positive rate; always records runtime and volume metrics; emits per-spider summaries for the Download service.

Operational notes.

- **Stateless vs. stateful**: DB Service is stateful (volume-backed); Diff and Download are stateless and horizontally scalable. Resource limits for Diff can be tuned independently.
- **Interfaces are stable**: DB exposes import/read views; Diff exposes run endpoints; Download serves artefacts. Backwards-compatible changes are guarded by schema and feature-version bumping.
- **Testing**: modules provide unit tests with fixtures; services have smoke tests (end-to-end `/run_diff/<name>`) and golden-sample validations for regression safety.

4.5.3 DB Service CLI Interface

A small internal CLI complements the HTTP API for quick admin and developer tasks without starting the full stack. It reuses the same code paths as the service, favors idempotent operations, and is meant for setup, diagnostics, and fast iteration.

- **Imports & refresh:** one-off ATP/OSM ingests and refreshing materialized feature views.
- **Inspect:** list tables, feature view versions, and recent run metadata.
- **Maintain:** clean transient staging tables and related housekeeping.
- **Dev utilities:** sample rows to CSV/GeoJSON and seed tiny test databases.
- **Execution & safety:** run via `docker compose run --rm db python cli.py` or locally (DuckDB); same config precedence as Section 4.5.4. Imports skip unchanged snapshots unless `--force`; migration guards prevent ad-hoc schema drift.

- **Out of scope:** diff generation, model training, and artifact export remain in their dedicated services to preserve auditing.

4.5.4 Configuration Management

The DiffedPlaces services are fully parameterized at runtime via a single, repository-tracked `config.json` that acts as the canonical source of truth for thresholds, feature toggles, data paths and integration behavior. The file is read at container start-up; validation is performed before any POI ingestion or OSM queries are executed to ensure safe, reproducible defaults.

Configuration layers and precedence. To keep deployments predictable yet flexible, configuration is resolved in four layers (earlier wins on conflicts):

1. *Built-in defaults* compiled into each service (defensive fallbacks only).
2. Repository `config.json` (project-wide canonical values).
3. Environment overrides (e.g., `DP_DIFF_DISTANCE_THRESHOLD_M=350`) injected by `docker-compose`.
4. Optional CLI flags for ad-hoc runs (e.g., `–spider ikea_ch`).

On boot, the service logs the effective configuration and *refuses to start* if the file is missing or invalid. Unknown keys trigger warnings to prevent silent drift.

Reloading and immutability. By default, configuration is immutable per run for reproducibility.

4.5.5 Testing Strategy

Unit Tests. Focus on deterministic components: normalization routines, string preprocessing, feature calculations, and rule-based match classification boundaries.

Integration Tests. Exercise a miniature dataset flowing through acquisition to diff generation, asserting invariants (non-empty diff, no duplicate identifiers, distance thresholds respected).

Golden Dataset Evaluation. Periodic model evaluation against curated labels; metric deltas outside tolerance trigger investigation.

Performance Checks. Sample timing of candidate generation and feature computation to detect accidental algorithmic regressions. However, performance was no high focus.

4.5.6 Test Data

Golden data includes manually validated POI pairs for selected brands/categories. Negative examples are diversified (geographically close yet distinct entities, name variants of unrelated brands) to discourage overfitting.

4.5.7 Error Handling and Logging

Structured logging captures run identifiers, configuration hash, and counts at each pipeline stage (records ingested, candidates generated, matches classified). Exceptions in external data fetching are retried with backoff. Ambiguous address parses are flagged rather than silently discarded.

4.5.8 Continuous Improvement Loop

Misclassified diff items discovered during manual review feed back into golden dataset updates, followed by retraining and re-evaluation, establishing a data-centric iterative refinement cycle. This feedback loop constitutes the primary agile mechanism: each cycle yields a potentially shippable improvement (more accurate matching or reduced ambiguity) validated in the subsequent biweekly review, enabling empirical prioritization over speculative planning.

4.6 Results and Further Development

This section summarizes achieved software artefacts and outlines prioritized enhancements.

4.6.1 Achieved Results

- Established modular pipeline implementing ingestion, normalization, candidate generation, matching, and diff export.
- Implemented baseline rule-based matcher producing actionable diff candidates with deterministic thresholds.
- Integrated supervised model(s) improving classification quality over rule-only baseline (relative F1 uplift over rule-only baseline reported in Section 3.7).
- Produced golden dataset generation workflow enabling iterative refinement.
- Added configuration-driven execution enabling reproducible runs across environments.
- Containerized deployment artefacts to reduce environment drift.

4.6.2 Qualitative Observations

Rule-based heuristics provide high precision on unambiguous brand/location clusters but underperform on dense urban clusters with overlapping categories. Machine learning features extracting nuanced textual and spatial interactions mitigate this gap. Data quality variance (address sparsity, coordinate noise) remains the dominant limitation to further gains. The iterative delivery approach ensured that each added capability (e.g., new similarity feature, additional retailer) was justified by observed error patterns in prior evaluation cycles rather than speculative completeness.

4.6.3 Limitations

- Limited diversity in golden datasets may bias model toward early-targeted brands.
- Lack of fully automated incremental update mechanism (full recomputation required).
- Absence of integrated human auditing UI (external tooling needed).
- Performance characteristics on nationwide multi-category runs not yet benchmarked systematically.

4.6.4 Future Enhancements

1. **Nationwide Spider Orchestration (CH)**: Operationalize regular, idempotent runs of all Swiss spiders with a unified scheduler (cron/CI), per-source health checks, delta crawling, and automatic retry/quarantine of failing sources. Define SLIs/SLOs (freshness, coverage, failure rate) and persist run manifests for reproducibility.
2. **Tight Coupling with OSM Auditor**: Replace one-off exports with a streaming handoff: stable identifiers, per-diff task creation, status sync (open/in review/accepted/rejected), and webhooks back to the pipeline when a review decision lands. Preserve reviewer context (screenshots, before/after geometry, confidence and features) to support transparent, auditable decisions.
3. **Throughput Improvements for ML-Based Matcher**: Reduce end-to-end ML-Based Matcher wall time for one Spider from $\sim$20 minutes to $\leqslant$1 minute (approximately 200 POIs) for the standard workload by (i) stronger spatial proximity filtering with category pre-filters, (ii) vectorized feature computation, (iii) batched inference, (iv) caching of expensive features in columnar stores (e.g., Parquet), and (v) parallel execution per canton/source. Track CPU time, peak memory, and cost per 1'000 candidates as primary KPIs.
4. **Model Quality & Rule-Base Revision**: Grow the golden set with active learning on high-uncertainty pairs. Recalibrate thresholds and re-tune the rule-based post-processor using the expanded gold labels; introduce an *abstain* path for low-confidence cases to route them to human review. Report per-category precision/recall and calibration error.

5. **Scalable Training Pipeline & Monitoring**: Decouple training-data generation from model training. Establish a dedicated pipeline with well-defined stages: (i) sampling/labeling and hard-negative mining, (ii) feature materialization to a versioned store, (iii) sharded training on large datasets, (iv) evaluation against fixed public/private gold sets, and (v) model registry with promotion rules. Provide experiment tracking (hyperparameters, seeds, metrics), resource/throughput monitoring, and drift alerts; schedule retraining jobs and dashboards for training time, examples/sec, and cost per 1'000 examples.
6. **Object-Oriented Refactor of `osm_conflate`**: Rework the command-line oriented implementation into a small library with clear components (*Reader* → *Blocker* → *Matcher* → *Resolver* → *Writer*). Use dependency injection for swappable strategies (distance metrics, scorers, conflict resolvers), preserve current OSM profile compatibility via thin adapters, and add unit/integration tests around each interface.
7. **Multi-DB Access & Concurrency**: Address the current single-writer/reader limitation of DuckDB by separating compute from persistence: stage intermediate results to Parquet/Arrow, use a single writer process (job queue) with many concurrent readers, or migrate transactional parts (task metadata, audit state) to a server database with snapshot isolation. Provide a connection/transaction policy that guarantees consistency without stalling parallel jobs.
8. **Smarter Incremental Runs**: Add snapshot differencing and tile-level invalidation so that only affected geographic cells are recomputed when sources or *osm_extracts* change. Maintain per-source content hashes and bitemporal stamps (ingest time vs. source time) to avoid redundant work and to enable precise replays.

4.6.5 Sustainability and Community Adoption

Clear documentation, transparent configuration, and minimal external dependencies increase the likelihood of adoption by the wider OSM community. Prioritizing reproducibility and traceability builds trust in generated diffs, a prerequisite for community-driven validation. Furthermore, the lightweight agile cadence (biweekly inspection of diff quality and metric deltas) reduced risk of large, late-stage integration surprises and accelerated convergence on stable matching thresholds.

4.7 Project Management

The project focuses on developing and generalizing advanced POI matching methodologies. Building on PT1 and PT2, the following structured timeline guides delivery, integrating the algorithms into the DiffedPlaces framework and validating their performance.

4.7.1 Phased Approach

The work is divided into five indicative phases. In practice, boundaries were porous and several tasks overlapped intentionally to enable earlier feedback (e.g., partial feature engineering began while generalization refactoring was still ongoing). This reflects an agile, inspection-driven cadence rather than a rigid stage-gate model.

Phase 1: Review of the literature and concept refinement (January)

- Conduct an in-depth review of current geospatial data integration and conflation techniques.
- Define evaluation metrics such as precision, recall, and F1 score to measure algorithm performance.
- Refine the conceptual framework, using the findings from PT1 and PT2.

Phase 2: Generalization of the DiffedPlaces Framework (February–March)

- Generalize the existing DiffedPlaces framework based on insights and implementations from PT1 and PT2.
- Adapt the framework to support all ATP spiders, ensuring compatibility with various data formats and enabling applicability across multiple regions and sources.
- Refactor the existing workflows to standardize the generation of *diff* files, improving scalability and modularity for further enhancements.
- Validate the generalized framework to ensure seamless integration with the existing pipeline and datasets.

Phase 3: Algorithm Development (March–May)

- Implement a rule-based algorithm that combines string-based comparison and geospatial proximity checks.
- Develop advanced algorithm utilizing machine learning techniques (Random Forest).
- Modularize the algorithms as a standalone Python package.
- Integration in the DiffedPlaces service.

Phase 4: Dataset Preparation and Testing (June)

- Prepare datasets by combining high-quality OSM data with external sources.
- Generate synthetic datasets to simulate both matched and unmatched POI pairs.

- Conduct performance benchmarks to refine algorithms for accuracy and efficiency.

Phase 5: Validation and Delivery (July–August)

- Validate the integrated algorithms against the prepared datasets and benchmarks.
- Ensure that the DiffedPlaces service operates efficiently under diverse data and geographic conditions.
- Document the final results and deliver the modular service for public use.

4.7.2 Timeline and Milestones

The initial high-level plan served as a directional guide; actual completion dates for some deliverables shifted modestly as early empirical findings reprioritized effort (e.g., extending time for candidate generation optimization reduced later rework in ML feature tuning). External evidence or formal tracking tools were intentionally minimized; the authoritative progress record resided in the outcomes demonstrated during biweekly meetings. For Milestones see 4.1.

Phase	Task	Delivery Date
Phase 1: Literature Review and Concept Refinement	Complete literature review and define performance metrics and refine conceptual framework (based on PT1/PT2)	26 January 2024
Phase 2: Generalization	Integrate algorithms, automate diff generation, generalize for all ATP spiders	09 March 2024
Phase 3: Rule-based Matching	Develop and integrate rule-based algorithm	20 April 2024
Phase 3: ML Matching	Develop and integrate ML-based algorithm	01 June 2024
Phase 4: Dataset Preparation and Testing	Prepare and validate datasets for benchmarking	22 June 2024
Phase 5: Validation and Delivery	Final validation of integrated algorithms	20 July 2024
Phase 5: Validation and Delivery	Deliver and document final results	10 August 2024

Table 4.1: Timeline and Milestones for Project Execution

4.7.3 Risk Management

Risks were surfaced informally during review meetings and addressed within the next iteration cycle; no separate risk register was maintained. Key risks and mitigation strategies are as follows:

- **Data Quality Issues**: Addressed through rigorous preprocessing and validation pipelines using trusted datasets.
- **Integration Challenges**: Ensured by modular development and extensive unit testing during integration into the DiffedPlaces service.
- **Performance Bottlenecks**: Managed through algorithm optimization and the use of efficient data processing tools like DuckDB.

4.7.4 Governance and Communication

Given the single-developer setting, governance emphasizes transparency of decisions and reproducibility of experiments over formal approval gates. Biweekly (every two weeks) stakeholder check-ins constituted the sole official communication artefact; no external evidence repositories, ticketing dashboards, or burndown charts were produced. Key decisions (e.g., threshold adjustments, model retraining triggers) were captured implicitly in configuration diffs and model artefact versioning.

4.7.5 Tooling and Infrastructure

Core project management enablers (kept deliberately lightweight) include:

- **Version Control**: Git repository maintaining code, configuration, and golden dataset artefacts (branching strategy kept lightweight: main only).
- **Issue Tracking**: Within the meeting notes.
- **Automation**: Containerized environment definitions (docker-compose) to standardize local execution and reduce setup variability.

4.7.6 Change Control

Parameter changes (e.g., distance thresholds) and model updates were adopted only after quick comparative evaluation on the golden dataset; this fast feedback cycle replaced formal change approval. Rollback is supported by retaining prior model artefacts and configuration snapshots.

4.7.7 Quality Integration

Quality assurance is embedded continuously rather than deferred: unit tests validate normalization and feature engineering, while golden dataset evaluations guard against performance drift. Risk mitigation actions (e.g., expanding negative examples to counter overfitting) are integrated into the operational backlog rather than treated as incidental tasks.

4.8 Project Monitoring

Given the nature of the DiffedPlaces project as a one-person initiative with a single stakeholder, project monitoring was straightforward and focused on regular communication to ensure alignment and progress. The monitoring strategy relied on consistent meetings and open communication channels. Furthermore, the defined milestones were tracked during the project.

4.8.1 Regular Meetings

The project started with regular bi-weekly check-ins with the initiator/advisor. Around the midpoint, a domain expert joined and the cadence shifted to every 3–4 weeks to enable deeper reviews. Each session focused on progress, blockers, and decisions, with concise minutes and action items to keep delivery on track.

4.8.2 Focus on Work Priorities

During each meeting, the focus was on identifying the key tasks for the upcoming period and discussing any adjustments needed based on the evolving understanding of project requirements. This iterative review process allowed for flexibility and ensured that the project continued to progress efficiently toward its objectives.

4.8.3 Conclusion

Project monitoring remained intentionally lightweight yet effective due to the narrow stakeholder set and clear milestone structure. The cadence of bi-weekly reviews coupled with explicit artifact-based progress indicators (produced diff samples, evaluation metric snapshots, updated golden datasets) mitigated scope drift. Future scaling to a multi-contributor context would require formalizing additional metrics (lead time, code coverage, model drift alerts) and perhaps adopting a lightweight agile board. A concise internal metrics log (lead time per feature, diff review turnaround, weekly precision/recall snapshot) was maintained instead of a separate summary table. This minimalist approach mirrors agile principles: prioritize direct inspection of working software and empirical quality signals over maintaining secondary tracking artefacts that do not directly contribute to validated learning.

4.9 Software Documentation

This section serves as a navigational and maintenance guide to the DiffedPlaces codebase hosted at: `https://gitlab.com/geometalab/diffedplaces`. It complements earlier architectural (Section 4.4), implementation (Section 4.5), and requirements (Section 4.2) discussions without repeating them.

4.9.1 Repository Overview

The repository is organized along the pipeline boundaries already described conceptually in Section 4.4. Only a concise directory map is given here; detailed per-component responsibilities are documented in Section 4.5.

```
db/             # Data ingestion API, schema, candidate + feature SQL views
diff/       # Orchestrated matching + diff export (Rule-Based + ML-Based Matcher)
download/       # Lightweight UI/API to browse produced diff artefacts
lib/            # Shared domain models, normalization + similarity utilities
devtools/       # One-off analytical scripts (category mapping, diagnostics)
diagrams/       # Architecture and data flow diagrams (referenced in text)
tests/          # Unit + integration tests (distributed fixtures)
docker-compose  # Multi-service local orchestration configuration
config.json    # Canonical configuration (thresholds, feature toggles, paths)
```

4.9.2 Module Responsibilities (Cross-Reference)

For rationale and deeper behavior see Sections 4.4 and 4.5; only succinct pointers are listed here:

db/ Import endpoints (`/import_osm`, `/import_atp`), unified POIs schema management, spatial/blocking views, feature materialization.

diff/ Pipeline execution: candidate retrieval, feature retrieval, Rule-Based classification, ML-Based Matcher inference, aggregation, export of `result.json`, `.osm` diff, inspection GeoJSON, metrics.

download/ Read-only presentation of produced artefacts with provenance (run fingerprint, counts, evaluation metrics where available).

lib/ Deterministic utilities: string/address normalization, token similarity (e.g., Jaro–Winkler, token ratios), geodesic distance, category mapping helpers.

devtools/ Exploratory scripts supporting category mapping validation and data quality analysis; not part of the production execution path.

4.9.3 Configuration Reference

Central configuration mechanics (precedence, immutability per run, fingerprinting) are defined in Section 4.5.4. The file `config.json` groups: (i) distance and textual similarity thresholds, (ii) feature enable/disable flags, (iii) model bundle identifiers, (iv) source inclusion filters (e.g., enabled spiders), (v) export paths.

4.9.4 Execution Workflow

The concrete runtime sequence maps directly onto the abstract batch flow shown in Figure 4.1: acquisition (db service) → candidate + feature materialization → Rule-Based (or ML-Based) classification → aggregation → export → metrics logging. Orchestration is performed via HTTP endpoints or scripted invocations; container composition wires shared volumes for data, exports, and persistent DuckDB state. Repetition of step internals is intentionally avoided here; see Sections 4.4 and 4.5.

4.9.5 Primary Data Artefacts

- **Canonical POI Tables**: Unified schema instances with stable identifiers and source attribution (produced by `/import_osm` and `/import_atp`).
- **Feature Views**: Deterministic DuckDB views keyed by candidate pair identifiers encapsulating spatial, textual, and categorical features.
- **Diff Exports**: A JSON summary (`result.json`) and an `.osm` ChangeXML (see glossary entry ChangeXML) ready for auditing tooling; accompanied by matched/unmatched GeoJSON for spatial inspection.
- **Metrics**: Precision, recall, F1-Score, volume counts, runtime timings, and configuration fingerprint.
- **Golden Dataset**: Curated labeled pairs (refer to Analysis Section 4.3 for creation process) feeding evaluation and (future) ML training.

4.9.6 Extensibility Path

The procedural steps for adding new spiders, external OGDs sources, countries, or features are elaborated in the extensibility mechanisms subsection of the Design (Section 4.4). This documentation section defers to that source to avoid divergent instructions.

4.9.7 Deployment and Operations

Service boundaries, ports, statefulness, and compose wiring are described in Section 4.5. Operationally the `db` service is the only stateful component (DuckDB files + imported snapshots); `diff` and `download` are stateless and horizontally scalable. External scheduling (e.g., CI, cron) triggers runs; no embedded scheduler exists by design.

4.9.8 Maintenance Practices

Sustainable upkeep of the current solution relies on the following practices, documented where applicable in the service `README.md` files (`db/`, `diff/`, `download/`) and in the Copilot instructions:

- **Release and branching:** Semantic versioning with tagged releases; short `CHANGELOG` notes per release; clear branch conventions (e.g., `main` stable, `feature/*` for work in progress).
- **Dependency management:** Strictly pin Python packages, perform regular updates (monthly/bi-monthly), and scan container images for vulnerabilities; provide commands and workflow in the service `README.md`.
- **Consistent runtime:** Align on a single Python version across all services; use containers as the reference environment (`docker compose up`). Offer a `Makefile` (e.g., `make dev`, `make test`, `make build`).
- **Configuration hygiene:** Describe `config.json` fields (units, defaults, constraints) and validate at startup against a JSON Schema; list category-specific thresholds and examples in the `README.md`.
- **Data refresh cycles:** Define cadence for OSM/ATP refresh, schedule regular runs of the Swiss spiders, and set artifact retention rules (inputs, diffs, models) with a traceable directory structure documented per service.
- **Testing and CI:** Run `pytest` with a minimum coverage gate, lint/format with ruff/black, and type-check with mypy; automate via CI (build, test, image scan) and provide pre-commit hooks.
- **Observability:** Emit structured JSON logs with correlation IDs; expose health and readiness endpoints; track basic metrics (job duration, candidate count, final match rate). Include a lightweight runbook in the `README.md`.
- **Security hygiene:** Keep base images current, scan dependencies, minimize exposed ports, and pass API keys/secrets via environment variables (never commit them). Add a short checklist in the `README.md`.
- **ML maintenance:** Separate training-data generation from model training; version datasets and models with metadata; schedule periodic re-training and evaluation against golden sets; store confusion matrices and PR curves.
- **Matching quality rules:** Maintain category-specific distance thresholds; normalize and cache strings; validate taxonomy mappings regularly. Ship changes as small PRs with test cases.
- **Recovery and backups:** Back up critical artifacts (DuckDB files, models, configuration). Document restore steps in a “Recovery” section of the `README.md`.
- **Copilot instructions:** Provide prompt snippets for code style, tests, and performance hints; update guidance whenever module interfaces or conventions change.
- **Documentation currency:** For each functional change, update the relevant `README.md` with config implications and operational notes; consider Architecture Decision Records (ADR) for major choices.

Chapter 5

Conclusion and Outlook

5.1 Conclusion

This thesis addressed the conflation of heterogeneous OGDs and the community-driven OSM database by designing, implementing, and empirically evaluating a modular pipeline centred around the DiffedPlaces service. Building on the lineage and insights of [12, 13], the work advanced three complementary fronts: (i) a refined semantic blocking taxonomy and supporting tooling reducing spurious candidate pairings; (ii) a comparative study and iterative enhancement of rule-based versus supervised POI matchers; and (iii) an extensible software architecture that integrates data acquisition, feature computation, model execution, diff generation, and (planned) human auditing.

The research synthesis (Section 2.4.1) established that high-quality POI conflation benefits from hybrid spatial–semantic strategies and systematic evaluation. These principles guided the solution design (Chapter 3). The high-level category derivation and feedback loop (Section 3.4) measurably reduced reliance on an over-used fallback class (“other”), increasing precision of downstream similarity computations without compromising recall. This semantic blocking, combined with distance constraints, provided a computationally tractable candidate space while maintaining coverage across diverse domains.

Two matcher families were assessed: a grid-searched rule-based configuration (evolution of the PT1/PT2 heuristic engine) and a supervised Random Forest Classifier. The rule-based approach demonstrated strengths in interpretability, rapid execution (Table 3.8), and predictable tuning behaviour around similarity thresholds of 0.85–0.875 (Section 3.7.3). However, its precision–recall balance remained sensitive to sparse or noisy attributes and required manual curation of weights and comparators.

The machine learning (ML) matcher progressed from an initially high-precision but recall-deficient model (Phase 1 synthetic-only training) to a substantially improved Phase 2 model through semi-supervised augmentation (Section 3.7.4). Augmenting synthetic perturbations with carefully filtered pseudo-labelled pairs closed the recall gap and achieved near-symmetric, very high precision and recall on the Aldi Süd

CH golden subset. The comparative view (Section 3.7.6) underscores that data diversity and realistic variation are critical: synthetic perturbations alone did not capture brand naming, temporal drift, or subtle address inconsistencies observed in real data. While the ML model delivered superior effectiveness metrics (notably lower false positives and near-elimination of false negatives), it incurred markedly higher runtimes, highlighting an efficiency gap for future optimisation.

Architecturally, the DiffedPlaces service facilitated encapsulation of conflation stages, enabling focused iteration (e.g., swapping matchers) without destabilising ingestion or diff generation layers. The decision to integrate osm_conflate with minimal adaptation accelerated early diff production and leveraged an established representation familiar to experienced OSM contributors. Nonetheless, this lightweight inclusion imposed compatibility constraints (e.g., data model translation steps, limited leverage of enriched feature vectors) and duplicated some filtering logic already present upstream. A bespoke conflation writer tightly coupled to the enriched candidate and feature pipeline could reduce transformation overhead, enable finer-grained provenance annotations, and simplify auditing integration. The trade-off therefore balances time-to-value through reuse against longer-term maintainability and optimisation potential.

Overall, the thesis demonstrates that (1) systematically engineered semantic blocking materially improves downstream matching quality; (2) a supervised ensemble matcher, when supplied with heterogeneous and partially pseudo-labelled training data, can surpass a finely tuned heuristic baseline; and (3) modular service design expedites experimentation while preserving a clear path toward operational auditing and validation tooling. Limitations remain in runtime efficiency of the ML path, incomplete automation of auditing workflows, and partial reliance on external tools whose internal evolution may introduce upstream drift. These limitations inform the outlined outlook.

5.2 Outlook

Future work concentrates on four complementary trajectories: model efficiency, auditing robustness, data governance (temporal and provenance-aware), and architectural simplification.

Model efficiency and scalability. The current ML inference layer exhibits substantially higher runtime than the rule-based matcher (Table 3.8). Targeted optimisations include: (i) feature pruning guided by permutation importance to remove low-contribution similarity signals; (ii) distillation of the Random Forest Classifier into a gradient-boosted or shallow ensemble variant with comparable discrimination but reduced depth; and (iii) batch-oriented vectorisation and caching of reusable token normalisations. Profiling could further identify I/O versus CPU bottlenecks and inform container resource allocation. A medium-term objective is a latency envelope that narrows the performance gap while preserving precision and recall.

Auditing integration and human-in-the-loop workflows. Completing and hardening the planned auditing component (e.g., integration with an OSM Auditor-style interface) will operationalise safe data contribution. Priority enhancements include incremental diff review queues, fine-grained provenance (original attribute tokens, comparator scores), and reviewer feedback capture to seed active learning loops.

Temporal and brand-specific dynamics. Time-dependent divergences (e.g., store openings, relocations, or spider schedule misalignment) surfaced during exploratory diff generation (e.g., Lidl Switzerland test run with newly observed spider errors and temporal data inconsistencies). Formalising snapshot versioning—storing retrieval timestamps, spider commit hashes, and transformation rule versions—would support temporal diffing and root-cause analysis of mismatches driven by stale or premature data. Extending the golden dataset with additional stratified brand subsets (beyond Aldi Süd) can validate generalisability and detect category-specific bias.

Architectural refinement. Reassessing the light-touch inclusion of osm_conflate is warranted. A dedicated diff writer leveraging native feature representations might (i) eliminate intermediate format conversions, (ii) expose richer confidence and feature attribution metadata directly in the output, and (iii) simplify maintenance by reducing complexity. A staged migration path could run both implementations in parallel, benchmarking functional equivalence and community acceptance before deprecating the legacy path.

Monitoring, profiling, and container ergonomics. Debugging and monitoring inside containers proved cumbersome. Introducing structured logging, metrics export (e.g., request latency, match distribution, feature computation time), and lightweight tracing would accelerate diagnosis of bottlenecks. Automated profiling profiles (CPU, memory) during representative workloads can reveal optimisation opportunities, while a dedicated developer profile container (for on-demand brand-specific reprocessing) streamlines iterative experimentation.

Active learning and continuous improvement. Once an auditing interface yields labelled confirmations and rejections, an active learning loop can prioritise low-confidence or disagreement-prone candidate pairs for manual review, steadily enriching the training corpus with high-value examples. Periodic drift detection on category distributions and feature value ranges would trigger re-training or mapping adjustments when deviations emerge.

Governance and reproducibility. Versioned configuration bundles (category mappings, comparator settings, model hyperparameters) and immutable training artefacts (hash-addressable datasets) would strengthen reproducibility claims and facilitate longitudinal benchmarking across future iterations. Integrating automated regression evaluation (precision, recall, F1-Score, and runtime) into the deployment pipeline will guard against silent performance regressions.

In summary, the project establishes a robust foundation for automated, high-quality conflation of POIs between OGD and OSM. Advancing toward production-grade reliability entails deepening auditing capa-

bilities, optimising ML runtime, enriching temporal handling, and potentially replacing minimal external tool adaptations with purpose-built components tailored to the enriched feature pipeline.

Glossary

cf_audit Command line auditing tool shipped with the DiffedPlaces conflator 13, 36, 37

ChangeXML XML format understood by JOSM for applying bulk edits 13, 14, 61

CLI Command-line interface enabling direct invocation of selected application functions without a graphical user interface vii, viii, 27, 51, 76

conflation The process of merging two or more geospatial datasets into a unified representation 6, 7, 65, 70, 75

DiffedPlaces A service designed to synchronize and reconcile data from various sources with OpenStreetMap 7, 12, 37–39, 49, 52, 55–59, 63, 64

DuckDB An in-memory SQL database designed for high-performance querying of large datasets 13, 15, 37, 58, 75

F1-Score A metric used to evaluate the performance of a classification model, calculated as the harmonic mean of precision and recall, balancing false positives and false negatives 13, 32, 33, 38, 39, 61, 65

GeoJSON is a format for encoding a variety of geographic data structures using JavaScript Object Notation to represent simple geographical features, along with their non-spatial attributes. 15

JOSM Editor JOSM (Java OpenStreetMap Editor) is a desktop application for editing OpenStreetMap data. It is a powerful tool for advanced mapping tasks and provides features like aerial imagery support, plugins, and comprehensive tagging capabilities, enabling users to contribute detailed and accurate geographic information to the OpenStreetMap database. 12, 13, 75

KNN k-Nearest Neighbors, a machine learning algorithm used for classification and regression 6

LightGBM A gradient boosting framework that uses tree-based learning algorithms for classification and ranking 6, 39

OSM Auditor A web-based platform for collaborative auditing of diffs in OpenStreetMap (OSM) data. It enables users to review, validate, and manage differences between datasets, ensuring data accuracy and reliability. 38, 40, 41, 54, 65

OSM Conflator is a script tool for merging points from some third-party source with OpenStreetMap data. 12, 36

osm_conflate A repository containing tools and utilities for conflating external geospatial data with OSM data. It provides scripts and configurations, including cf_conflate, to facilitate the comparison, merging, and validation of data sets to ensure accuracy and consistency in OSM. [15] 12, 37, 64, 65, 71, 75

Overpass API is a specialized query language for extracting data from the OpenStreetMap database. 75

PT1 The previous project, which focused on a matching algorithm utilized for improving the accuracy of Point of Interest (POI) matching. PT1 provides a generic approach to data matching, making the integration of new external data sources easier. 55, 56

PT2 The follow-up project to PT1, which enhanced the matching algorithm by introducing generalization techniques. PT2 extended the algorithm's capabilities to handle diverse datasets and improve scalability for broader data integration. 55, 56

Random Forest Classifier A machine learning method for classification, regression, and other tasks that operates by constructing multiple decision trees 11–13, 22, 24, 27, 29, 32, 34, 37, 63, 64, 77

TF-IDF Term Frequency-Inverse Document Frequency, a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents 6, 20, 21

VGI Volunteered Geographic Information, geospatial data provided by individuals voluntarily 5, 41, 75

Acronyms

ATP All The Places 3, 4, 7, 12–19, 21, 28–30, 35, 38, 39, 41–49, 51, 56, 57, 79, 80

CDIS Centralized Data Integration Service 8

ODbL Open Data Commons Open Database License 3, 13–15, 36, 40

OGD Open Government Data 1, 6–8, 10, 11, 14, 30, 47, 61, 63, 65, 76

OSM OpenStreetMap 1, 3–19, 21–24, 26, 28–30, 32, 34–36, 38–46, 48, 49, 51, 52, 55, 56, 63–65, 68, 75, 80, 81, 83

POI Point Of Interest 1–3, 5–8, 10–17, 19, 21, 24, 26, 27, 29, 30, 32, 35, 38, 39, 41, 42, 44, 52–56, 60, 63, 65, 70, 76

Appendix A

Personal Reflection

This reflection summarises personal learning outcomes, principal challenges encountered, mitigation strategies, and professional growth emerging from the project. It intentionally avoids introducing new technical contributions beyond those documented in the main chapters.

Learning Outcomes

The work deepened practical competence in orchestrating an end-to-end POI conflation workflow: data acquisition, semantic normalisation, feature engineering, matcher evaluation, and diff generation. A key insight was the compounding value of early, rigorous taxonomy curation; disciplined semantic blocking reduced downstream ambiguity and clarified error provenance. Exposure to both heuristic and supervised matcher paradigms reinforced an evidence-based approach to model selection: empirical precision–recall behaviour, rather than architectural novelty, guided adoption decisions.

Software engineering skills matured through designing modular service boundaries (ingestion, matching, diff writing) that supported iterative substitution (e.g., swapping matcher implementations) with controlled risk. Emphasis on reproducibility—structured logs, versioned mappings, and golden datasets—highlighted its role in sustaining credible evaluation claims. Managing experimental artefacts (metric tables, configuration grids) cultivated systematic record-keeping habits beneficial for future applied research.

Key Challenges and Mitigations

Heterogeneous data quality. Inconsistent or sparse attributes (addresses, opening hours) complicated heuristic weighting. Mitigation: down-weighting empirically unstable fields and enriching training data with pseudo-labelled examples to diversify pattern coverage.

Synthetic–real gap. Early ML training on purely synthetic perturbations produced over-optimistic precision and underwhelming recall on real brand data. Mitigation: semi-supervised augmentation with filtered high-precision rule-based matches and rigorous hold-out of golden sets.

Temporal drift and spider variability. Test diffs (e.g., Lidl Switzerland) surfaced time-dependent discrepancies and spider-specific extraction errors. Mitigation: manual inspection, logging of snapshot timestamps, and scoping a roadmap for explicit temporal provenance capture.

Container debugging ergonomics. Introspecting runtime performance and defects within containers was cumbersome. Mitigation: ad hoc profiling, structured log augmentation, and planning for dedicated developer-oriented runtime profiles and future observability tooling.

Tool integration trade-offs. Minimal adaptation of osm_conflate accelerated early output but introduced conversion overhead and constrained richer attribution emission. Mitigation: clear articulation of a potential migration path toward a native diff writer, preserving short-term velocity while outlining long-term simplification.

Use of AI-Assisted Tools

AI tools (notably conversational assistants and code completion systems such as ChatGPT and GitHub Copilot) were leveraged for: (i) rapid scaffolding of boilerplate code or configuration templates; (ii) synthesising alternative phrasings during documentation drafting; and (iii) sanity-checking regular expressions and data transformation snippets. Their effective use required tight human verification loops: generated suggestions were validated against project-specific constraints (naming conventions, glossary terms, citation correctness) to avoid subtle inconsistencies or hallucinated references. The exercise reinforced disciplined prompt formulation and critical evaluation of AI-produced output, underscoring that productivity gains hinge on domain oversight rather than unconditional acceptance.

Professional Growth and Perspective

The project strengthened capacity to balance experimentation with operational pragmatism: deciding when to invest in architectural generality versus delivering a functional baseline for empirical comparison. Exposure to semi-supervised model evolution cultivated a nuanced view of data curation as an iterative, feedback-driven asset rather than a static prerequisite. Finally, deliberate integration of reproducibility practices and reflective error analysis established habits directly transferable to future applied data engineering or geospatial integration initiatives.

Future Personal Development

Future growth areas include deeper optimisation of model serving performance, expansion of active learning methodologies for human-in-the-loop refinement, and more systematic observability (metrics, tracing) to shorten diagnostic cycles. Continued refinement of AI tool usage—especially for test generation and exploratory data summarisation under strict validation—is expected to compound efficiency while maintaining quality.

Overall, the project provided a comprehensive environment for advancing technical, evaluative, and reflective competencies central to robust, data-intensive system development.

Appendix B

Declaration of Independence

I hereby declare that I, Claudio Bertozzi, have completed the present master's thesis independently and without unauthorized assistance. All sources used in this thesis, including those quoted or paraphrased, have been acknowledged and cited according to academic standards.

Furthermore, I acknowledge that I have utilized AI tools, such as ChatGPT by OpenAI, during the preparation of this thesis. These tools were used to support text formulation, idea development, and content review. The use of such tools was conducted in accordance with ethical guidelines and the regulations of my academic institution.

I also confirm that this work has not been submitted, either wholly or in part, for any other degree or examination at this or any other institution.

Location, Date: Stäfa, 11.08.2025

Signature:

Appendix C

Tooling and Platforms Utilized

This appendix enumerates the principal tools, platforms, and services employed throughout the project lifecycle. It complements the Declaration of Independence (Appendix B) by providing transparent disclosure of the technical and AI-assisted environment. The listing is organized by functional role (version control, authoring, design, data, collaboration, automation) and notes the specific contribution each tool made. Proprietary tools were used only where an open alternative would have imposed disproportionate overhead.

Version Control and Project Hosting

Git: Distributed version control for source, configuration, experiment artifacts, and LaTeX sources. Granular branching supported parallel exploration (e.g., matcher variants) and facilitated reproducible re-runs via commit pinning.

GitLab: DiffedPlaces repository

Source Authoring and Editing

Visual Studio Code: Primary editor for source code (Python, configuration, scripts) and centralized editing of LaTeX files, leveraging syntax-aware linting and integrated terminals.

Overleaf: Collaborative review of the main LaTeX thesis document, ensuring consistent formatting with the institutional class file while enabling asynchronous supervisory feedback.

Diagramming and Visualization

draw.io (diagrams.net): Creation of architecture, data flow, and configuration diagrams (e.g., system overview, conflation pipeline) exported to vector or optimized raster formats placed under version control.

Data Processing, Storage, and Query

DuckDB: In-process analytical querying for intermediate tabular feature inspection (cf. glossary entry DuckDB).

Geospatial and Conflation Tooling

osm_conflate: External conflation utilities leveraged for initial diff generation scaffolding (see osm_conflate).

JOSM Editor: Manual inspection and spot validation of selected diffs (see JOSM Editor).

Overpass API: Targeted extraction of OSM entities for test and validation cases (see Overpass API).

AI-Assisted Productivity

GitHub Copilot: In-line code completion and LaTeX phrasing suggestions, accelerating boilerplate generation (configuration templates, repetitive transformation code) under human review to prevent hallucinated identifiers.

ChatGPT: Conversational brainstorming for alternative matcher feature formulations, explanatory text drafts, and refinement of section structuring. All outputs were critically validated; no uncited external claims were accepted without corroboration. (Declaration of usage: Appendix B).

Research and Literature

Google Scholar and Google: Discovery of background literature on geospatial conflation, volunteered geographic information (VGI), and machine learning similarity measures; bibliographic metadata cross-checked against maintained entries in the project `bibliography.bib`. No uncited search results were included.

Automation and Experiment Support

Shell Scripting / CLI: Reproducible invocation of ingestion, feature computation, and evaluation routines through parameterized scripts; ensured consistent environment variable handling and artifact placement.

Ethical and Quality Safeguards

Human oversight remained mandatory for AI-suggested content, particularly to enforce glossary term consistency (e.g., POI, OGD) and citation validity. Any autogenerated proposal lacking a verifiable source was rejected or explicitly flagged for later verification.

Reproducibility Note

All substantive artifacts (code, configurations, diagrams, and generated feature lists) are under version control, enabling reconstruction of reported behaviors via commit references. Where transient exploratory tools were used without persistent outputs, they are disclosed above for completeness.

Appendix D

Category Encoding for ML Matching

This appendix documents the categorical one-hot / multi-level encoding referenced in Section 3.6.2. It is derived directly from the configuration (no external sources) and is used to create the combined tag features consumed by the Random Forest Classifier. For each OSM key listed, two tiers of encodings are generated:

1. *Main key presence (12 features)*: one feature per primary key (e.g., `amenity`, `shop`). Value meanings: 0 (absent in both), 1 (present in exactly one of the pair), 2 (present in both).
2. *Specific key:value presence (124 features)*: one feature per enumerated value (e.g., `amenity:restaurant`, `shop:supermarket`). Same 0/1/2 encoding.

The current configuration yields 136 categorical features (12 main, 124 specific). Together with the seven rule-based similarity components, two name presence flags, and two derived ratios this produces a feature vector length of 147.

Main Keys

`amenity`, `tourism`, `emergency`, `shop`, `office`, `highway`, `craft`, `leisure`, `aeroway`, `aerialway`, `healthcare`, `building`

Specific Key:Value Entries

`amenity:restaurant`, `amenity:cafe`, `amenity:bar`, `amenity:pub`, `amenity:fast_food`, `amenity:phar`
`amenity:hospital`, `amenity:clinic`, `amenity:doctors`, `amenity:dentist`, `amenity:school`,
`amenity:college`, `amenity:university`, `amenity:kindergarten`, `amenity:fuel`, `amenity:bus_stat`
`amenity:bus_stop`, `amenity:railway_station`, `amenity:taxi`, `amenity:parking`, `amenity:cinema`,
`amenity:theatre`, `amenity:sports_centre`, `amenity:park`, `amenity:bank`, `amenity:atm`, `amenity:li`

amenity:townhall, amenity:courthouse, amenity:fire_station, amenity:police, amenity:post_
amenity:post_box, amenity:hairdresser, amenity:car_repair, amenity:dry_cleaning,
amenity:veterinary, amenity:bicycle_rental, amenity:bureau_de_change, amenity:car_rental
amenity:car_wash, amenity:casino, amenity:charging_station, amenity:compressed_air,
amenity:fountain, amenity:kick-scooter_rental, amenity:money_transfer, amenity:music_ver
amenity:payment_terminal, amenity:recycling, amenity:toilets, amenity:vending_machine,
amenity:bicycle_parking, amenity:motorcycle_parking, amenity:public_bookcase, amenity:wa
amenity:drinking_water, amenity:parking_entrance, tourism:hotel, tourism:hostel,
tourism:guest_house, tourism:motel, tourism:museum, tourism:artwork, tourism:attraction,
tourism:camp_site, tourism:apartment, tourism:information, tourism:picnic_site, emergency
emergency:defibrillator, emergency:phone, emergency:disaster_help_point, emergency:life_
shop:convenience, shop:bakery, shop:hairdresser, shop:supermarket, shop:car_repair,
shop:clothes, shop:florist, shop:kiosk, shop:farm, shop:sports, shop:butcher, shop:beauty,
shop:bicycle, shop:shoes, shop:optician, shop:furniture, shop:electronics, shop:jewelry,
shop:car, shop:mall, shop:department_store, shop:doityourself, shop:mobile_phone,
shop:copyshop, shop:perfumery, shop:interior_decoration, shop:dry_cleaning, shop:pet,
shop:hardware, shop:gift, shop:newsagent, shop:toys, shop:deli, shop:tattoo, shop:books,
shop:chemist, shop:pharmacy, shop:cosmetics, office:government, office:company, office:insu
office:financial, office:consulting, office:estate_agent, office:it, highway:bus_stop,
highway:traffic_signals, highway:crossing, highway:turning_circle, highway:parking

Usage in Feature Vector

For each pair (source, OSM) the encoding assigns 0/1/2 as described. This compact ternary representation replaces a larger set of separate one-hot indicators (e.g., separate binary features for each side) and allows the classifier to learn symmetric relations ("both have" vs. "only one has") with a single split. Category signals supplement, rather than replace, lexical and geometric similarities.

Appendix E

Data Sources: Catalog and Taxonomy

This appendix summarizes the ATP spider identifiers used and the high-level category taxonomy referenced in Section 3.3. It supports provenance and repeatability without introducing implementation details.

E.0.1 ATP spider identifiers

The following list enumerates the ATP spider identifiers used for Swiss outlets. Identifiers correspond to the upstream data sources configured in the project.

accor - activ_fitness_ch - adidas - adlo - aesop - alamo - aldi_sud_ch - alstom - american_vintage - anicura - apple - audi - avis - bauhaus_ch - bern_ch - best_in_parking - bmw_group - boconcept - boels - bonita - bottega_veneta - bp - burger_king - c_and_a - calvin_klein - carhartt_wip - casino - century_21 - claires - clever_fit - coffee_fellows - coinstar - coop_vitality_ch - credit_agricole - dean_and_david - decathlon_ch - deichmann - denner_ch - depot - discover_swiss - dm - dominos_pizza_ch - douglas_ch - dunkin_ch - eathappy - ecco - ernst_young - exxon_mobil - eyes_and_more - fairmont - fielmann_ch - fit_active - five_guys_ch - fjallraven - foot_locker - ford - fressnapf_ch - fust_ch - g_star_raw - gant - gbfs - gifi - global_sample_spider_200ch - google_offices - grandvision - guess - hard_rock - hermes - hertz - hilton - hm - hollister - hooters - hugo_boss - hyundai_ch - ich_tanke_strom - ikea - ikks - intermarche - ionity - joe_and_the_juice - jumbo_ch - just_over_the_top - jysk - kate_spade - kaufland - kik - kiko_milano - koerperformen - landi_ch - lego - levis - lhw - lidl_ch - little_free_library - livique_ch - losteria - lovisa - lush - madame_frigo_ch - man_truck_and_bus - marionnaud - marriott_hotels - maserati - mcdonalds - medbase_ch - mephisto - mercedes_benz_group - meta -

migros_ch - misenso_ch - mitsubishi_ch - mol - moncler - moneygram - montblanc - mueller - national - natuzzi - nespresso - newyorker - nh_hotel_group - nike - nissan - nkd - nordsee - notfalltreffpunkte_ch - old_wild_west - omegawatches - optical_center - ottos_ch - our_airports - pandora - pokawa - porsche - prada - pret_a_manger - pricewaterhousecoopers - puma - ralph_lauren - rebel_architette - renault - revolution_laundry - rituals - ritz_carlton - rotpunkt_apotheken_ch - rsg_group - scania - seat - shell - sixt - skechers - skoda - societe_generale - soeder_ch - sostrene_grene - stadt_zuerich_ch - storebox - suitsupply - sushi_daily - swing_kitchen - tag_heuer - tedi - tesla - the_body_shop - tiffany - timberland - toyota - travelex - trek_bikes - undiz - update_fitness_ch - valora - vans - versace - volg_ch - volkswagen - volvo - warhammer - waterdrop - western_union - winterthur_ch - worldcat - wycon_cosmetics - wyndham

E.0.2 Category overview

The high-level category taxonomy used for harmonising ATP categories and OSM tags comprises the following classes:

- **gastro**: Food and drink establishments like restaurants, cafes, bars, and fast food.
- **health**: Healthcare facilities including hospitals, clinics, pharmacies, and medical practices.
- **education**: Educational institutions like schools, universities, and kindergartens.
- **transport**: Transportation-related facilities like fuel stations, parking, and public transport.
- **leisure**: Entertainment and leisure venues like cinemas, theaters, parks, and sports centers.
- **finance**: Financial services including banks and ATMs.
- **public**: Public services and government buildings.
- **service**: Service providers like hairdressers, repair shops, and professional services.
- **retail**: Retail stores and shopping establishments.
- **accommodation**: Lodging facilities like hotels, hostels, and camp sites.
- **other**: Points of interest that do not fit into other categories.

E.0.3 Snapshots and sources

For reproducibility, record the specific inputs used in this study:

- **ATP snapshot dates:** *6. June 2025*

- **OSM extract:** *Geofabrik, Switzerland, 6. June 2025*
- **Boundary dataset:** *Swiss Country boundaries GeoJSON, commit/date 05. Februar 2025*

Glossary

cf_audit Command line auditing tool shipped with the DiffedPlaces conflator 13, 36, 37

ChangeXML XML format understood by JOSM for applying bulk edits 13, 14, 61

CLI Command-line interface enabling direct invocation of selected application functions without a graphical user interface vii, viii, 27, 51, 76

conflation The process of merging two or more geospatial datasets into a unified representation 6, 7, 65, 70, 75

DiffedPlaces A service designed to synchronize and reconcile data from various sources with OpenStreetMap 7, 12, 37–39, 49, 52, 55–59, 63, 64

DuckDB An in-memory SQL database designed for high-performance querying of large datasets 13, 15, 37, 58, 75

F1-Score A metric used to evaluate the performance of a classification model, calculated as the harmonic mean of precision and recall, balancing false positives and false negatives 13, 32, 33, 38, 39, 61, 65

GeoJSON is a format for encoding a variety of geographic data structures using JavaScript Object Notation to represent simple geographical features, along with their non-spatial attributes. 15

JOSM Editor JOSM (Java OpenStreetMap Editor) is a desktop application for editing OpenStreetMap data. It is a powerful tool for advanced mapping tasks and provides features like aerial imagery support, plugins, and comprehensive tagging capabilities, enabling users to contribute detailed and accurate geographic information to the OpenStreetMap database. 12, 13, 75

KNN k-Nearest Neighbors, a machine learning algorithm used for classification and regression 6

LightGBM A gradient boosting framework that uses tree-based learning algorithms for classification and ranking 6, 39

OSM Auditor A web-based platform for collaborative auditing of diffs in OpenStreetMap (OSM) data. It enables users to review, validate, and manage differences between datasets, ensuring data accuracy and reliability. 38, 40, 41, 54, 65

OSM Conflator is a script tool for merging points from some third-party source with OpenStreetMap data. 12, 36

osm_conflate A repository containing tools and utilities for conflating external geospatial data with OSM data. It provides scripts and configurations, including cf_conflate, to facilitate the comparison, merging, and validation of data sets to ensure accuracy and consistency in OSM. [15] 12, 37, 64, 65, 71, 75

Overpass API is a specialized query language for extracting data from the OpenStreetMap database. 75

PT1 The previous project, which focused on a matching algorithm utilized for improving the accuracy of Point of Interest (POI) matching. PT1 provides a generic approach to data matching, making the integration of new external data sources easier. 55, 56

PT2 The follow-up project to PT1, which enhanced the matching algorithm by introducing generalization techniques. PT2 extended the algorithm's capabilities to handle diverse datasets and improve scalability for broader data integration. 55, 56

Random Forest Classifier A machine learning method for classification, regression, and other tasks that operates by constructing multiple decision trees 11–13, 22, 24, 27, 29, 32, 34, 37, 63, 64, 77

TF-IDF Term Frequency-Inverse Document Frequency, a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents 6, 20, 21

VGI Volunteered Geographic Information, geospatial data provided by individuals voluntarily 5, 41, 75

Acronyms

ATP All The Places 3, 4, 7, 12–19, 21, 28–30, 35, 38, 39, 41–49, 51, 56, 57, 79, 80

CDIS Centralized Data Integration Service 8

ODbL Open Data Commons Open Database License 3, 13–15, 36, 40

OGD Open Government Data 1, 6–8, 10, 11, 14, 30, 47, 61, 63, 65, 76

OSM OpenStreetMap 1, 3–19, 21–24, 26, 28–30, 32, 34–36, 38–46, 48, 49, 51, 52, 55, 56, 63–65, 68, 75, 80, 81, 83

POI Point Of Interest 1–3, 5–8, 10–17, 19, 21, 24, 26, 27, 29, 30, 32, 35, 38, 39, 41, 42, 44, 52–56, 60, 63, 65, 70, 76

List of Figures

List of Tables

References

[1] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[2] W. E. Winkler, "String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage," *Proceedings of the Section on Survey Research Methods, American Statistical Association*, pp. 354–359, 1990.

[3] C. Mülligann, K. Janowicz, M. Ye, and W. Lee, "Analyzing the spatial-semantic interaction of points of interest in volunteered geographic information," in *Spatial Information Theory (COSIT 2011), Proceedings of the 10th International Conference, Belfast, ME, USA, September 12–16, 2011*, ser. Lecture Notes in Computer Science, vol. 6899. Springer, 2011, pp. 350–370.

[4] G. Giannopoulos, K. Alexis, N. Kostagiolas, and D. Skoutas, "Classifying points of interest with minimum metadata," in *ACM SIGSPATIAL International Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising (LocalRec)*. Chicago, IL, USA: ACM, 2019, pp. 1–4.

[5] X. Xing, F. Zhao, H. Lin, and S. Qiang, "Local poi matching based on knn and lightgbm method," in *2nd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*. IEEE, 2022, pp. 455–460.

[6] K. Sun, Y. Hu, Y. Ma, R. Zhou, and Y. Zhu, "Conflating point of interest (poi) data: A systematic review of matching methods," *Computers, Environment and Urban Systems*, 2023.

[7] A. Kashian, K.-F. Richter, A. Rajabifard, and Y. Chen, "Mining the co-existence of pois in openstreetmap for faulty entry detection," in *3rd Annual Conference of Research@Locate*, Melbourne, Australia, 2016, pp. 1–8.

[8] M. Piech, A. Smywinski-Pohl, R. Marcjan, and L. Siwik, "Towards automatic points of interest matching," *ISPRS International Journal of Geo-Information*, vol. 9, no. 5, p. 291, 2020.

[9] R. Low, Z. D. Tekler, and L. Cheah, "An end-to-end point of interest (poi) conflation framework," *ArXiv*, vol. abs/2109.06073, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237491615

[10] H. Bast, P. Brosi, and M. Näther, “Similarity classification of public transit stations,” in *ACM SIGSPATIAL*, 2020.

[11] Zverik, “Osm conflator,” https://wiki.openstreetmap.org/wiki/OSM_Conflator, 2024.

[12] C. Bertozzi, “Matching and conflation of open government data with openstreetmap data,” OST – Eastern Switzerland University of Applied Sciences, Tech. Rep., 2024.

[13] ——, “Matching and conflation of open government data with openstreetmap data,” OST – Eastern Switzerland University of Applied Sciences, Tech. Rep., 2024.

[14] DuckDB, “Duckdb documentation,” https://duckdb.org/docs/, 2024, accessed: 2024-08-27.

[15] mapsme, “osm_conflate,” https://github.com/mapsme/osm_conflate, 2024.

[16] O. Knowledge, “Open data commons open database license,” https://opendatacommons.org/licenses/odbl, 2024.

[17] A. T. P. Community, “All the places,” https://wiki.openstreetmap.org/wiki/All_the_Places, 2024.

[18] SeatGeek, “Thefuzz,” https://github.com/seatgeek/thefuzz, 2023.

[19] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[20] scikit learn, “scikit-learn,” https://github.com/scikit-learn/scikit-learn, 2023.